TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

IINES PIESALA

# CONTENT BASED VISUALIZATION OF LINKED ONLINE DATA

Master of Science thesis

Examiner: prof. Tommi Mikkonen
Examiner and topic were approved by the Faculty Council of the Faculty of Computing and Electrical Engineering Council meeting on 6.5.2015.

# ABSTRACT

**IINES PIESALA**: Content based visualization of linked online data
Tampere University of technology
Master of Science Thesis, 49 pages, 2 Appendix pages
November 2015
Master's Degree Program in Information Technology
Major: Software Engineering
Examiner: Professor Tommi Mikkonen

Keywords: search, graph, web application, data, visualization

The amount of data has grown exponentially since the first introduction of Web. The search for specific data from the Internet is ever more difficult because of the amount of data the web holds. Search engines offer a way to search the Internet based on keywords. Search results are shown as a list of links to most relevant websites. The search engines search and present well responses to specific keywords. However these search engines do not handle questions about overall picture of a keyword well.

This master's thesis researched an idea how to enable the search of whole ecosystem and showing the result to the user. The search would only take simple a user request as input. An example story was used throughout this thesis. This story was about user, who wanted to see all technology conferences visualized through connections of speakers and sponsors of technology conferences. This visualization would have enabled the user to see at glance how many sponsors and speakers the technology conferences have in common. The goal of this master's thesis was either to find existing application or build own implementation, which would solve the problem previously described.

The proposed systems theory is based on genre recognition of a website, web search, named-entity recognition (NER) and graphs. Based on the scientific literature these subjects were presented and discussed. There were no existing application that would have worked as mentioned before. The other option was to implement the proposed system, this option was chosen. The prototype application is a mix of own implementation and external APIs. These APIs were used to search the web and recognize named-entities. The prototype application was implemented as web application, which used web technologies such as JavaScript and Node.js.

The prototype application was tested with a case study. The case study used the technology conference example mentioned above. The results of the prototype application were compared to manually acquired data from five technology conference websites. 82% of the technology conferences found by the prototype application were real technology conferences. Based on the results the speakers were more recognized than the sponsors. However the sponsors were more accurately recognized. Only few of the sponsors in the result graph were not actual sponsors of the conferences. The resulting graph had more false speakers than false sponsors.

The prototype application proved the idea successful. However the prototype application did not meet the initial plan of general usage. The technology conference case study showed the potential of the idea. Still further research and work is needed to utilize the full potential of the prototype application.

# TIIVISTELMÄ

**IINES PIESALA**: Sisältöön perustuvan netistä löytyvän linkittyneen tiedon visualisointi
Tampereen teknillinen yliopisto
Diplomityö, 49 sivua, 2 liitesivua
Marraskuu 2015
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Tommi Mikkonen

Avainsanat: haku, graafi, web-sovellus, data, visualisointi

Tiedon määrä Internetissä on kasvanut räjähdysmäisesti sen syntymän jälkeen. Tiedon etsiminen on yhä vaikeampaa tiedon määrän vuoksi. Internetin hakupalvelut hakevat nettisivuja käyttäjän hakusanojen perusteella ja näyttävät listan tuloksista. Nämä palvelut ovat hyviä hakemaan tarkkaa tietoa, mutta pärjäävät huonommin kokonaisuuksien hakemisessa ja esittämisessä.

Tässä diplomityössä tutkittiin ideaa, miten mahdollistaa kokonaisuuksien etsiminen ja näyttäminen käyttäjälle pelkästään käyttäjän syötteen perusteella. Esimerkkinä oli käyttäjän toive nähdä kaikki teknologiakonferenssit, joissa esiintyy jokin seuraavista avainsanoista: HTML5, Node.js, JavaScript, ja niiden puhujat ja sponsorit. Käyttäjä halusi nähdä kuinka paljon samoja sponsoreita ja puhujia eri teknologiakonferensseilla on. Tämän diplomityön tavoitteena oli joko löytää olemassa oleva sovellus tai tehdä oma sovellus, joka ratkaisisi tämän ongelman.

Edellä kuvattu sovellus nojautuu vahvasti nettisivun genren tunnistamiseen, Internethakuun, nimettyjen asioiden tunnistamiseen ja graafeihin. Näihin tutustuttiin useiden tieteellisten artikkeleiden avulla. Etsinnöistä huolimatta täysin vastaavaa sovellusta, joka olisi toteuttanut edellisessä kappaleessa kuvatun toiminnallisuuden, ei löytynyt. Näin ollen ainoaksi vaihtoehdoksi jäi toteuttaa prototyyppisovellus itse. Prototyyppisovellus käyttää haussa ja nimettyjen asioiden tunnistamisessa ulkopuolisia rajapintoja. Prototyyppisovellus on web-sovellus, joka toteutettiin käyttämällä webteknologioita, kuten JavaScriptiä ja Node.js:ää.

Prototyyppisovelluksen toimintaa testattiin Case-tutkimuksena. Case-tutkimuksessa käytettiin teknologiakonferenssi esimerkkiä. Prototyyppisovelluksen antamia tuloksia teknologiakonferenssi käyttäjäkyselylle verrattiin viiden eri teknologiakonferenssin osalta manuaalisesti hankittuihin tietoihin. Prototyyppisovelluksen antamista teknologiakonferensseista 82% oli oikeasti käyttäjän kuvaukseen sopivia teknologiakonferensseja. Tulosten perusteella prototyyppisovellus tunnisti puhujat paremmin kuin sponsorit. Lähes kaikki tulosgraafin näyttämät sponsorit olivat oikeita teknologiakonferenssin sponsoreita. Tulosgraafissa puhujien joukossa oli enemmän olemattomia puhujia, jotka eivät olleet listattuna teknologiakonferenssin puhujissa.

Prototyyppisovellus onnistui osoittamaan, että aiemmin kuvattu sovellus on mahdollista toteuttaa. Prototyyppisovelluksen yleiskäyttöisyys oli ainoa asia, mikä ei toteutunut toivotulla tavalla. Näiden tutkimustulosten perusteella voidaan sanoa, että idea prototyyppisovelluksen takana toimii.

# PREFACE

This thesis was written for Nokia Technologies. I want to thank my supervising Prof. Tommi Mikkonen for patience, great advice and encouragement. I want to express my gratitude to my manager Roope Kylmäkoski, with whom we brainstormed the idea behind the thesis. I would also want to thank my colleagues for support and push to finish the thesis. Special thanks to everyone commenting and proofreading the thesis.

I also want to thank my fiancé Aleksi for endless support and believing in me. You made me laugh and forget my stress, when I was stressed about the thesis. Finally I want to thank my family and friends for your support and patience. I would not be here without you. Special thanks to Maija-Leena and Susanna for listening and giving me words of encouragement.

Tampere, 21.10.2015

Iines Piesala

# CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

The amount of data has grown exponentially since the first introduction of Web. The search for specific data from the Internet is ever more difficult because of the amount of data the web holds. Search engines offer a way to search the Internet based on keywords. Usually only couple of keywords is used. Search results are shown as a list of links to most relevant websites. After the search result list is shown, it is left to the user to find the intended information from the websites. The search engines search and show well responses to specific keywords. However these search engines do not handle questions about overall picture of a keyword well. Google has implemented a feature called Knowledge graph to provide basic information about the searched entity ("Introducing the Knowledge Graph: Things, Not Strings" n.d.). For example a search for Ada Lovelace returns a picture of her and basic information about who she was. However this only works for single entities. There is no application to offer a simple way to search and view the connection between multiple entities and also show information about the entities.

This master's thesis researches an idea how to enable the search of whole ecosystem and showing the result to the user. The search would only take simple a user request as input. An example story is used throughout this thesis. This story is about user, who wants to see all technology conferences visualized through connections of speakers and sponsors of technology conferences. This visualization would enable the user to see at glance how many common sponsors and speakers the technology conferences have. The goal of this master's thesis is either to find existing application or build own implementation of the application. The application should be able to show ecosystem based on users input. The user input should not limited to any particular subject.

The structure of the thesis is explained next. Chapter 2 presents the motivation behind this master's thesis and researches the theory needed to implement the proposed system. Chapter 2 splits to five sections: motivation, website genre recognition, search, named-entity recognition and web graph. Prior art applications and application programming interfaces (APIs) were researched in chapter 3. Possibility of an own implementation was also discussed in chapter 3. The chapter 3 concludes the best way to build the proposed system. Chapter 4 presents the chosen implementation of the proposed system. Implementation of the proposed system is discussed through model-view-controller-pattern. Chapter 5 introduces the case study and its results. The case study uses the same technology conference example. Chapter 5 also evaluates how the prototype application succeeded and what is there to improve. Chapter 6 concludes the research results.

# 2. BACKGROUND

This chapter provides the theoretical background for the implementation of the system introduced in the first chapter. Section 2.1 describes the motivation of this thesis and describes the technology conference example. Section 2.2 describes the evolution of recognizing genre of website. Recognizing the genre of website helps with the filtering process and also finding the entities. Section 2.3 describes the history of search and search engines to illustrate how difficult is the process of serving quality results to the users. Section 2.4 introduces the named-entity recognition, its history and basic methods used to extract the entities from text. Extracting the entities from the websites is the very foundation of the system proposed in the introduction. Section 2.5 explains the basics of graph and how the graph structure exists also in the web.

## 2.1 Motivation

The web map is one of examples of visualization how web sites are connected. The web map presents the web sites as circles. Figure 1 presents the web map of Finland. The size of the circle corresponds to the amount of traffic every website. The proximity of different web sites on the map is derived from people' behavior, how people move from one site to another. If web sites are in close proximity, then users have frequently visited web sites via each other's links. The data visualized in the web map was from Alexa web service ("Alexa Top 500 Global Sites" n.d.). Alexa offers customers information about websites users, traffic, where the users came from to that particular site.

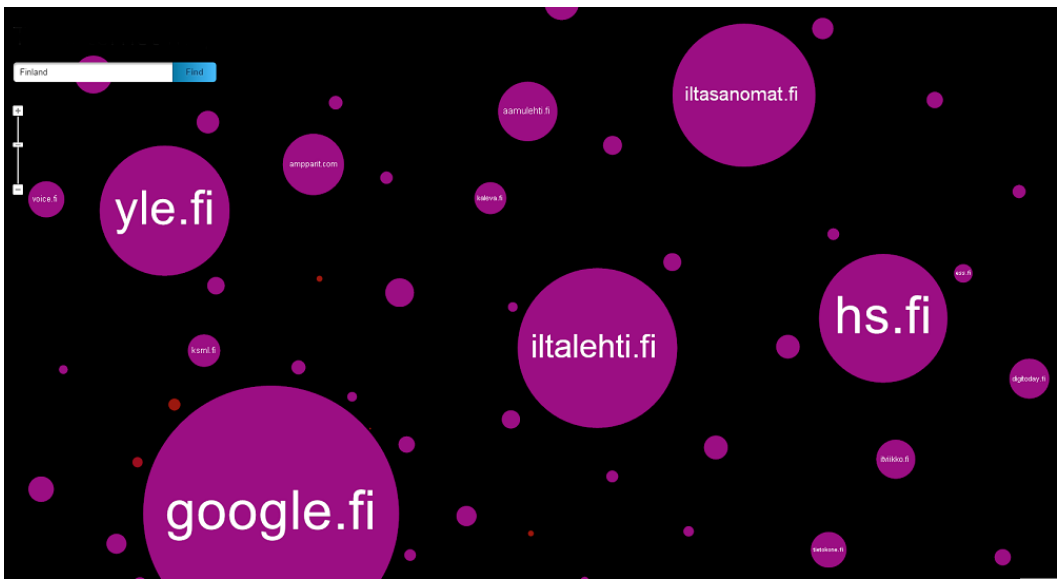


***Figure 1.*** *Finland Web Map.*

The web map makes the connection based on how the people have moved from one webpage to another. The movement between the websites most logically come from the hyperlinks.

This master's thesis describes the process of how the intended system would work. The system can be divided into four phases as seen in Figure 2.

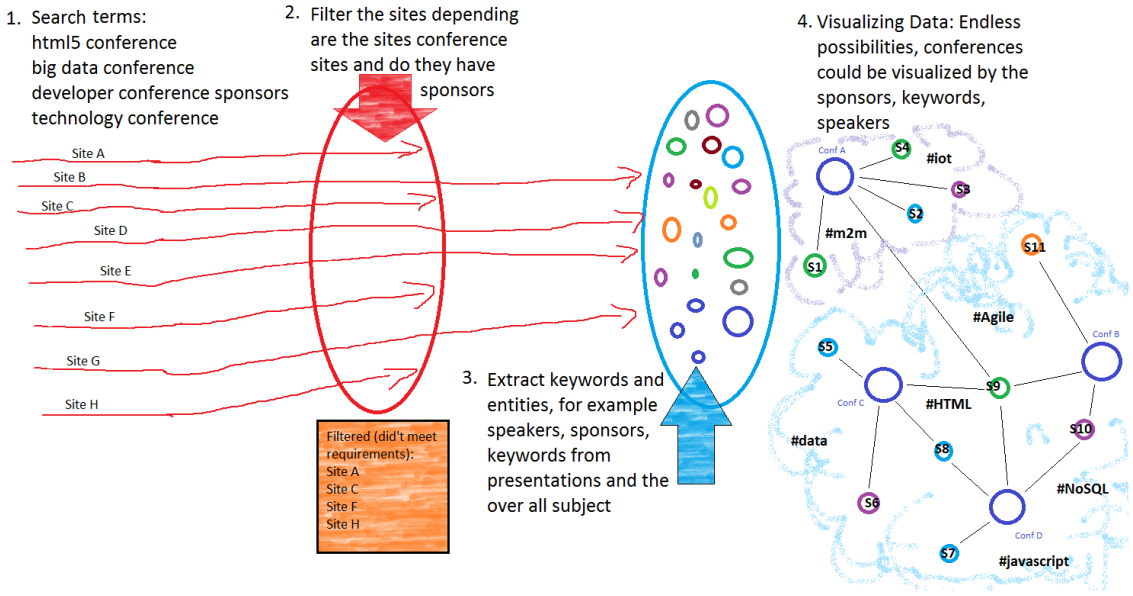


***Figure 2.*** *Technology conference example of the proposed system.*

Phase 1 consist of the users requests, what information she/he wants to visualize and explore. User request in Figure 2 was "I want to see how conferences are linked via sponsors, speakers and technologies. The chosen conferences should have sponsors and speakers. The conference subject should be big data, HTML5, developer or technology. From this request the system should be able to recognize the wanted information and do searches in the web accordingly. In this case the system would recognize that it should gather information from conference web sites on speakers, sponsors and key technologies. Only conference sites that have sponsors on display would qualify to have their information gathered.

Phase 2 system filters the search results based on the qualifications mentioned before. For example the filter should be able to determine whether website is in fact a conference homepage and also has sponsors. After the filter has filtered the unwanted results, the result set is ready for Phase 3.

In Phase 3 the system should be able to gather only the wanted information from the conference webpage. According to the Figure 2 the system will gather the basic conference information, such as name, location and time of the event, speakers, sponsors and key technologies, which are discussed in the conference. After the Phase 3 the wanted information should be stored in a structured way in a database.

Phase 4 will visualize the information saved in the database. User should be able to decide, which way data is visualized. For example in Figure 2 user has chosen to view the conference information based on sponsors and the link between sponsor and conference

describes the sponsorship. The picture above also shows the keyword clouds, which inform the user what kind of keywords were present in the conference website.

## 2.2 Recognizing genre of website

In 1992 Yates and Yorlikowski defined genre in the following way : "Genres are typified communicative actions characterized by similar substance and form and taken in recurrent situations." (Yates and Orlikowski 1992) Figure 3 shows three different research studies about genres of web.

| | Reproduced and emergent genres of communication on the World Wide Web | Genre Classification of Web Pages | **An Exanimation of genre attributes for web page classification** |
|---|---|---|---|
| **Genres** | 48 | 8 | 4 |
| **Sample size** | 100 | 800 | 1280 |
| **Genre classification** | Manually | Automatically | Automatically |
| **Year of the study** | 1997 | 2004 | 2008 |

***Figure 3.*** *Genre studies.*

The Web was different back in 1997: only a fraction in the popularity and size compared to 2008. In 1997 websites were static, containing only text, and possibly pictures. The study conducted in 1997 found 48 genres from the sample of 100 web sites. This was a surprise even for the authors. Unlike in the other two studies, there were no pre-selected genres. The large amount of genres found was result of authors themselves assigning genre for each website. The assigned genres were as precise as possible, even if the website would have fitted in less specific genre. Genres were also more tied to traditional genres found in literature than in the newer studies. Book, report, newsletter, concert review and filmography were some of the examples of traditional genres. (Crowston and Williams 1997)

"People who search the World Wide Web usually have a clear conception: They know what they are searching for, and they know of which form or type the search result ideally should be."(Eissen and Stein 2004) The technology conference example is based on this premise. When the user searches for the conferences, he/she knows the genre of the website is a conference homepage not news site. Another key takeaway is that not only the users know what content they are searching but also the form of the searched website. In the case of the conference backstory, the user would also know the conference homepage usually has list of speakers and sponsors. "64% of the students think that genre classification is very useful, and that another 29% find it sometimes useful …" (Eissen and Stein 2004) This conclusion also supports the premise that using genre as a search term is somewhat useful.

Using genre to classify the webpage is a good way to filter unwanted results, which otherwise would be included in the results. This would ease the filtering in the technology conference case, because the wanted information should be strictly from conference webpages not from for example news sites. In this context portrayal genre means web appearances of companies, universities and institutions (non-private) and private self-portrayals (private). Following these principles technology conference homepages genre is non-private portrayal. According to Figure 4 non-private portrayal webpage classification performance was 57,9%.

| | Shop | Portrayal (priv) | Portrayal (non-priv) | Article | Link Collection | Help | Discussion | Download | total |
|---|---|---|---|---|---|---|---|---|---|
| Shop | 66.9% | 3.0% | 11.2% | 5.3% | 7.7% | 3.0% | 1.2% | 1.8% | 100.0% |
| Portrayal (priv) | 0.0% | 67.7% | 3.1% | 8.7% | 15.7% | 2.4% | 2.4% | 0.0% | 100.0% |
| Portrayal (non-priv) | 7.0% | 4.1% | 57.9% | 5.8% | 18.1% | 2.9% | 2.3% | 1.8% | 100.0% |
| Article | 0.0% | 1.6% | 3.3% | 81.3% | 8.1% | 3.3% | 0.8% | 1.6% | 100.0% |
| Link Collection | 0.5% | 4.4% | 10.8% | 11.3% | 67.6% | 1.0% | 3.4% | 1.0% | 100.0% |
| Help | 2.2% | 2.2% | 5.1% | 19.1% | 10.3% | 55.1% | 2.2% | 3.7% | 100.0% |
| Discussion | 2.4% | 0.0% | 3.1% | 5.5% | 7.1% | 7.9% | 68.5% | 5.5% | 100.0% |
| Download | 2.0% | 1.3% | 5.9% | 5.3% | 2.5% | 1.3% | 2.0% | 79.6% | 100.0 % |

***Figure 4.*** *Ten-fold cross-validated confusion matrix.* (Eissen and Stein 2004)

Usually the portrayal (non-private) genre was mixed with shop and private portrayal, because there is lot of variance in non-private portrayal webpages. For example JSConf EU ("JSConf EU 2015" n.d.), Nokia ("Nokia" n.d.) and Tampere University of Technology ("Tampere University of Technology" n.d.) homepages vary greatly in content and form. Mixing the link collection and non-private portrayal may be because of surprisingly high number of links found in portrayal (non-private) webpages. Company homepages often describe their products in detail, which could explain why the homepages are mixed with shops.

Back in 2008 the web had evolved from only static websites to much more dynamic web sites. Rise of Flash, video and JavaScript made the static sites look like full blown desktop applications. Identifying genres automatically is challenging, because webpages constantly evolve and number of their genres rises. Next paragraphs before next chapter are combination of authors prior experiences and study called "An examination of genre attributes for web page classification." (Dong et al. 2008) . The research studied the genre classification with only four genres: Personal Home Page, FAQ, E-shopping and News. Genres were chosen for their distinguishability. The main point was figuring out how different combinations of genre attributes affected the genre classification. There were three genre attributes: content, form and functionality. Functionality describes what the user can do on the web page. Couple of examples of functionality are the scripts and

attributes. Figure 5 explains the characteristics of each genre attributes for each chosen genre.

| Web Genre | Content | Form | Functionality |
|---|---|---|---|
| **Personal Home Page** | Information about the site owner | Hierarchy Info. of related sub-topics | Scroll, emails, links to subtopics |
| **FAQ** | Pairs of Q. and A. | List | Scroll, search, links |
| **E-shopping** | List of products & services with details | Hierarchy | Scroll, search, emails, online inquiry and ordering |
| **News** | Multi-media items, poll, chat forum, News items | Hierarchy time-stamp | Dynamic info, navigation. links, search, login, multi-media on-site play, survey |

***Figure 5.*** Typical characteristic of content, form and functionality for each genre type. (Dong et al. 2008)

Machine learning was the approach used to identify the genres automatically. The data set contained 1280 web pages, which included 170 instances of each of four genres and random set of 600 web pages as noise data set. Figure 6 summarizes the mean precision and recall for each genre. According to Figure 6, automatic genre classification is able differentiate successfully among different types of genre.

| Genre | Precision | Recall |
|---|---|---|
| **E-Shopping** | 0.920 (0.075) | 0.902 (0.073) |
| **FAQ** | 0.992 (0.023) | 0.894 (0.088) |
| **News** | 0.978 (0.930) | 0.987 (0.018) |
| **PHP** | 0.863 (0.073) | 0.939 (0.048) |

***Figure 6.*** *Mean precision and recall for genre. Standard deviations in parenthesis.* (Dong et al. 2008)

PHP (Personal Home Page) had the worst precision of the four genres. The technology conferences homepage is part of PHP genre, when chosen from these four genres. FAQ and News genres have much more formalized content, form and functionality, which makes classification easier. The same study also researched the importance of combining genre classification attributes. Those results are presented in Figure 7. As seen in Figure 7, it is better to use a combination of attributes rather than classify the genre according only one attribute.

| Attribute | Precision | Recall |
|---|---|---|
| Content | 0.905 (0.110) | 0.838 (0.172) |
| Form | 0.928 (0.084) | 0.926 (0.078) |
| Functionality | 0.938 (0.063) | 0.944 (0.065) |
| Content & Form | 0.941 (0.067) | 0.952 (0.060) |
| Content & Functionality | 0.947 (0.065) | 0.931 (0.114) |
| Form & Functionality | 0.955 (0.056) | 0.959 (0.047) |
| Content & Form & Functionality | 0.952 (0.058) | 0.965 (0.043) |

***Figure 7.*** *Mean precision and recall for attribute type. Standard deviations in parenthesis.*(Dong et al. 2008)

It may be surprising to see that combining all three attributes does not make the automatic classifier perform significantly better than combining only two attributes. However the recall is the best when three attributes are used for genre classification.

When searching and identifying conference sites automatically, at least two attributes should be used. Probably the best solution is to use all three attributes content, form and functionality, because it is relatively easy to name features from each of those attributes. Content of a conference homepage is information about the conference, speakers, sponsors, venue and schedule. The form of conference homepage is a hierarchical information about related sub-topics. The functionality of a conference homepage is for example html tag names mentioning sponsor or speaker, links to company homepages and images of speakers and sponsors.

## 2.3 Search

MOT Oxford Dictionary of English ("MOT Oxford Dictionary of English" n.d.) gives the word "search engine" the following meaning: "a program that searches for and identifies items in a database that correspond to keywords or characters specified by the user, used especially for finding particular sites on the Internet". The database mentioned in the description in the case of Internet search engine is not a catalog maintained by officials because the sheer amount of web sites and their updates is really hard and costly to maintain. One of the most known search engines on the planet at the moment is Google

("Google" n.d.). The search engine laid the foundation for its success story by differentiating from the other search engines, at that time, with better ranking system for webpages. The commercial search engines use web crawlers to crawl the Internet. Search is relevant for the proposed system, because it is easier to filter out irrelevant websites with search. For example in the technology conference example conference websites are easier to find, if search is available. Without search, the whole Internet should be crawled in search of technology conference homepages.

### 2.3.1 Web crawler

Web crawler is a program that searches the Internet to create index (database). This description was provided by the MOT Oxford Dictionary of English. Web crawlers were also researched in "WWW Robots and Search Engines" research paper written by Heinonen, Hätönen and Klemettinen in 1996. This subsection is based on the "WWW Robots and Search Engines" research paper (Heinonen, Hätonen, and Klemettinen 1996). In 1996 the Internet had exploded from being a small medium mainly used by the academia to a large medium accessible also to large public audience. The web crawlers are the answer to finding the right information from the vast amount of available information in the Internet. Web crawlers can also be used to get and save information from Internet to use it later. This is the use case for the technology conference case.

The paper (Heinonen, Hätonen, and Klemettinen 1996) describes three different use cases for the robots also known as web crawlers, which are resource discovery, mirroring and link maintenance. Resource discovery is the main use case for web crawlers today, because it means the robot will crawl and index the Internet. The resources collected from the visited web pages depends on the robot, some robots collect only some summarizing information and some collect the whole web pages, which usually are broken into index of occurrence of words. The database of words is usually used by the search engines.

Using robots to mirror the web pages from one continent to another is not as relevant today as it was back in 1996, when the Internet connection was not as fast as it is today. During the 1996 period it was convenient to have a copy of the web page on multiple continent, so accessing the web pages was faster and easier with slow Internet connection. The third application for robots is link maintenance. The robots find easily the dead links in web pages, because they go through the link structure continuously. Although, as the authors also note, the dead links are not such a big problem and nowadays it is even smaller problem, because search engines nowadays produce quality search results. Often the search engine is the first waypoint to a specific web page not the home page.

The research paper (Heinonen, Hätonen, and Klemettinen 1996) described shortly the ethics problems with using robots to crawl the web. The authors briefly introduced the standard for robots exclusion guidelines, which the robots should always follow. The guidelines of the standard for robots exclusion describe robots.txt file structure, where the

admin of the web page can determine what pages/folders can be crawled and which robots have the right to crawl the web page. Those guidelines are important because robots increase server loads and also require bandwidth.

It was pondered how to measure the quality and usefulness of the search engine. The main aspects raised by them were the effectiveness of retrieval, information up-to-dateless, fastness of the search engine and index structure effectiveness. As the list shows almost all of the attributes of useful and good quality search engine are quantitative. The research concentrates to Alta Vista search engine and praises the fastness of the search engine. Alta Vista search engine had a special ability to restrict searches to certain portions of documents. For example user could have searched web pages that had university in their title. The ranking of the search result was done "… according to the appearance of the query terms in the first few words of the document, their appearance of the query terms in the first few words of the document, their appearance close to each other and their frequencies in the document. ". Basically the ranking system relied only on textual information. This was one of the reasons which led to the invention of PageRank ranking algorithm, because the textual information did not provide good enough metrics about the quality of the search result.

### 2.3.2 PageRank

Paper "The PageRank Citation Ranking: Bringing Order to the Web" (Page et al. 1998) researched the ways to define the "importance" of web page by using the link structure of the Web. The research paper was written by Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd in 1998. This subsection is based on that research paper. During that time the Internet was a lot smaller and it was not as widespread as it is today. It is hard to imagine searching for "Tampere University Of Technology" without expecting the homepage of the university be at least in top three search results. Back in 1998 world of web page ranking in search result was different according to the authors of the paper. This was well described above with the Alta Vista search engine, whose ranking system based heavily on ranking according to textual information.

One of the attempts to determine the relevance of web page was the amount of backlinks. Backlink is an incoming link to a web page. The idea of using backlinks as the measurement of quality of webpages came from the academic world where the amount of citations is often used as measurement of quality of the research paper. In this case the websites backlinks were thought as citations. The authors discussed the problems of only using the backlink count as quality measurement. One of those problems was the possible manipulation of the amount of the backlinks. The research also concluded that the backlink counting does not correspond to people's common sense notion of importance.

The research paper (Page et al. 1998) used as an example Yahoo's home page at that time (1998), according the research had 62 804 backlinks, which was exceptional amount,

because generally web page had only few backlinks in their research material. If a web page has a link off the Yahoo home page, it is more important than a web page that has more links but the links come from less popular web pages.

PageRank is a rating method, which rates the websites mechanically and objectively using backlink counts and calculating also the importance of every backlink. It is based on the graph of the web. Figure 8 from the original research paper demonstrates well the PageRank calculation process. Figure 8 is also a good demonstration of the Yahoo home page example, where the importance of where the link is coming from is taken into account.

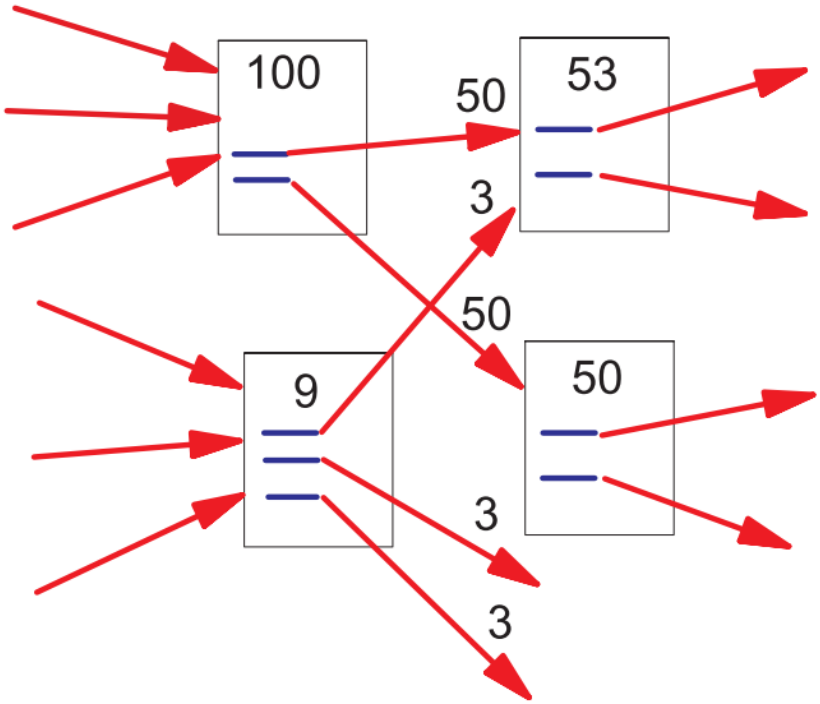


***Figure 8.*** *Simplified PageRank Calculation.* (Page et al. 1998)

The exact mathematical equation and implementation of PageRank is out of scope of this thesis. The research paper (Page et al. 1998) describes the crawling process how the web pages were collected and also the PageRank calculation process. According the research paper "The benefits of PageRank are the greatest for underspecified queries." Search query of Stanford University is good example of PageRank, because it would return the home page of the university as first search result and other conventional search engines would return other web pages. Those web pages have mentioned the university, without any notion about the importance and quality of the web pages. PageRank provides users with higher quality search results. (Page et al. 1998)

### 2.3.3 Google search engine

Sergey Brin and Lawrence Page are the authors of a research paper about the anatomy of a large-scale hyper textual web search engine (Brin and Page 1998). This subsection is based on that research paper. The search engine they described is nowadays known as Google search engine. The Internet and the world of search engines was different in 1998, when this research paper was published. The main motivation for developing new kind of search engine was the lack of good automated search engines on the market. The quote from the research paper addresses this well "Automated search engines that rely on key-word matching usually return too many low quality matches".

In 1997 it was common that "junk results" washed out any relevant search results for the user. The authors stressed the importance of few first 10 search results, even though the amount of web pages has increased by many orders of magnitude. The Google search engine used two important features to produce high precision in the results, which are quality ranking for each web page (called PageRank as mentioned in "The PageRank Citation Ranking: Bringing Order to the Web"-paper (Page et al. 1998)) and anchor text. The authors of the research paper use anchor text because it may provide more accurate description about the web page than the web page itself.

Google search engine has also other features mentioned in the research paper such as location of all hits (words), style of the words (like font size, is word marked as bold), and it saves the full raw HTML of the web pages into its repositories. The search engine uses hit lists to list all occurrences of a particular word in a particular document including position, font, and capitalization information. Multi-word search is described as complicated in the paper. The best search result for the query has to take the proximity of the words into consideration in addition to finding each world from the hit lists. Results of the research paper show that Google search engine produced better results than the other major commercial search engines for most searches at the time of the research.

## 2.4 Named-entity recognition

Finding the relevant information from the wanted web pages can be done with help of named-entity recognition (NER). NER extracts people, places, and organizations that are mentioned in text by proper name (as opposed to being referenced by pronominal terms, e.g., 'you', or nominal forms e.g., 'the man'). (Campbell, Dagli, and Weinstein 2013) Named-entity recognition is important part of visualizing the technology conference information. Finding the sponsors and speakers from the conference homepage could be done by using NER. Technology conference homepages usually contain names of the speakers (people) and sponsors (organizations), which according to the definition is what NER does best. This section is based on paper "A survey of named entity recognition and classification" as a resource to discuss NER history and features.(Nadeau and Sekine 2006)

Named Entity Recognition was formed as byproduct of Information Extraction tasks, whose goal was to extract structured information about companies' activities. When doing the Information Extraction (IE) work people found out the importance of being able to recognize information units like names (person, organization, location), numeric expressions (time, date, money) and percentage expressions. The method identifying references to these entities was called "Named Entity Recognition and Classification (NERC)" and it was recognized as one of important sub-tasks of IE. Later the classification part was dropped from the term and it is nowadays usually called Named Entity Recognition (NER). The paper presented the short history of research in NERC field from 1991 to 2006.

One of the first research relied on heuristics and handcrafted rules to extract and recognize company names. The NERC field has studied multiple different languages, but English has been the most popular language. It is good to note that the language has an effect to NER because some of the entities are language or culture bound; for example German has different word capitalization rules than English. The textual genre or domain also has an impact to NERC. According the authors any domain can be reasonably supported, so NERC is not domain specific. The only problem is the transferability of the system, because porting system designed for one specific domain to another is challenging.

The first part of the term "Named Entity" restricts the task of recognizing entities to only those entities that can be described explicitly. In the history of NERC research the main problem was described as recognizing the "proper names". According to the article "overall the most studied types are three specializations of 'proper names': names of 'persons', 'locations' and 'organizations'". When entity fell outside of the previously described specializations the type of that entity was called "miscellaneous". There has been also couple of research, which also discuss the more fine grained subcategories.

The research paper also introduced three most often used features, which were used for recognition and classification of named entities, of NERC. Those three features were word-level, list lookup and document and corpus features. Figure 9 presents the subcategories of word-level features with examples of the use cases. Digit pattern can be used to present for example year with four or two digits. Digit pattern can also present dates, prices, percentages and intervals. Morphology studies the form of words and it is essentially related to words affixes and roots. Nationality words are good example of common endings in words "ish" and "an" (Finnish, Swedish, Russian). If system is given enough examples of the nationality words it may learn to associate human professions with "ish" and "an" word endings.

| Features | Examples |
|---|---|
| **Case** | - Starts with a capital letter<br>- Word is all uppercased<br>- The word is mixed case (e.g., ProSys, eBay) |
| **Punctuation** | - Ends with period, has internal period (e.g., St., I.B.M.)<br>- Internal apostrophe, hyphen or ampersand (e.g., O'Connor) |
| **Digit** | - Digit pattern<br>- Cardinal and Ordinal<br>- Roman number<br>- Word with digits (e.g., W3C, 3M) |
| **Character** | - Possessive mark, first person pronoun<br>- Greek letters |
| **Morphology** | - Prefix, suffix, singular version, stem<br>- Common ending |
| **Part-of-speech** | - proper name, verb, noun, foreign word |
| **Function** | - Alpha, non-alpha, n-gram<br>- lowercase, uppercase version<br>- pattern, summarized pattern<br>- token length, phrase length |

***Figure 9.*** *Word-level features.* (Nadeau and Sekine 2006)

The second feature mentioned by the research paper was list lookup features, which are presented in Figure 10. Lists make searching entities a lot easier, because those lists can be just searched to match any entities. It also provides the means to remove unnecessary content from the source text, like the stop words or common abbreviations. List of entities can also provide a hint of the structure and word-level features common in for example organization entities. List of entity cues provides this information without the list of entities and creating an algorithm to find those hints. For example phrase that has Inc. at the end of it, is probably a good candidate for organization entity.

| Features | Examples |
|---|---|
| **General list** | - General dictionary<br>- Stop words (function words)<br>- Capitalized nouns (e.g., January, Monday)<br>- Common abbreviations |
| **List of entities** | - Organization, government, airline, educational<br>- First name, last name, celebrity<br>- Astral body, continent, country, state, city |
| **List of entity cues** | - Typical words in organization<br>- Person title, name prefix, post-nominal letters<br>- Location typical word, cardinal point |

***Figure 10.*** *List lookup features.* (Nadeau and Sekine 2006)

The third feature the research paper mentioned was document and corpus features. Document features encompass both document content and its structure. Figure 11 presents the document features that are beyond the single word and multi-word expression. It also includes the meta-information about the documents. In the case of this thesis the most

interesting of the document features is the document meta-information. A webpage source code has lot of meta-information and structure information.

| Features | Examples |
|---|---|
| **Multiple occurrences Examples** | - Other entities in the context<br>- Uppercased and lowercased occurrences<br>- Anaphora, coreference |
| **Local syntax** | - Enumeration, apposition<br>- Position in sentence, in paragraph, and in document |
| **Meta information** | - Uri, Email header, XML section<br>- Bulleted/numbered lists, tables, figures |
| **Corpus frequency** | - Word and phrase frequency<br>- Co-occurrences<br>- Multiword unit permanency |

***Figure 11.*** *Features from documents.* (Nadeau and Sekine 2006)

## 2.5 Web graph

As the technology conference example described, graph could be used to visualize the links between conferences and their speakers and sponsors. This chapter describes the basics of graph theory. In addition graph visualization benefits are discussed later based on studies made on that field. History of graph goes back to Euler, who lived in the 18$^{th}$ century and had a problem, where he used to draw graph of the map and solve the problem. Although in this case the solution to the problem was that the problem was not solvable.

Written in layman terms a graph has finite amount of nodes together with a set of ordered or unordered pairs of these nodes. These pairs are called arcs.(Harary 1969) If the direction of the arcs is relevant, then the graph is called directed graph. Otherwise the graph is undirected, because the direction of the arcs does not matter. Figure 12 shows an example of what directed graph looks like. The circles are the nodes and the lines between the circles are the arcs. The web can also be seen as a graph, where the nodes are webpages and arcs are the hyperlinks between the webpages. (Donato et al. 2007) More precisely the web is rather a directed graph than undirected graph, because the hyperlinks have a direction. The hyperlink is shown on a webpage and it leads to another webpage, thus the direction is from the webpage it is shown to that webpage the hyperlink is linked. At least the PageRank ranking algorithm uses the web graph as its advantage to determine which webpages are relevant for the user based on the search query.

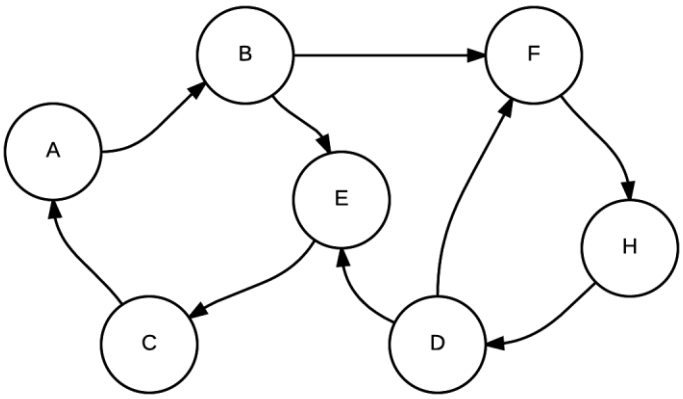


***Figure 12.*** *Directed graph.*

A graph can be visualized in many ways, but not all graphs are aesthetic, attractive or easy to understand. The algorithms for drawing graphs paper lists the main aesthetics, which are display symmetry, avoiding arc crossing, avoiding bend arcs, keeping arc lengths uniform and distributing nodes uniformly.(Purchase, Pilcher, and Plimmer 2012)

Showing the whole web graph on users display is impossible, because the size of the web graph is so huge and understanding such huge graph is difficult for the user. One research team used to show the sitemap as a graph to ease the users' navigation through the pages, but the large amount of links made the graph look messy and hard to read. The amount of information showed to the user has to be well thought, the graph should show only the relevant web graph effectively using clustering and layout adjustment and filter out the unnecessary data.

Layout of the web graph should make the picture easy to understand and remember. Most classical graph drawing algorithms produce aesthetically pleasing abstract graph layouts, only drawback to these algorithms is that the size of the node is expected to be small. Those algorithms are not designed to have any content inside the nodes. If those classical graph algorithms are used to show the web graph, either the nodes size should be relatively small and information should be showed outside the node or an alternative graph drawing algorithm should be used. (Lai, Technologies, and Huang 2010)

# 3. PRIOR ART AND ALTERNATIVES

This chapter introduces the prior art and alternatives. These could be used to make the proposed system reality. Section 3.1 describes the system requirements for the proposed system. Section 3.2 presents prior art of search, web scraping, NER and visualization. In section 3.3 the possibility of implementing prior art or a solution of the proposed system is discussed. Finally section 3.4 summaries the findings.

## 3.1 System requirements

The basics of the system requirements were lightly discussed in the technology conference case. This chapter presents the most important and basic requirements for the system to function well. The most basic requirement for the proposed system is access to Internet. Without Internet connection the proposed system will not work, because it fetches all the data used from the Internet. Also the faster the bandwidth the faster the proposed system can fetch the resources.

The proposed system is language dependent. It should only take into consideration material written in English. All non-English material found should be discarded, because language is relevant in NER. Results might be skewed, if non-English material would be included. User requests should also be in English.

Flow of system interactions with user is described next. First the user gets an idea, that she/he wants to see conferences and conference speakers and sponsors visualized in a graph. User writes the request on her/his device to the UI. Request is handled and after a while user gets a response, which shows the technology conference graph.

The most important takeaway from the story is the fluent and easy user interaction with the proposed system. User experience will not necessary be as fluent, if the proposed system would require combination of multiple applications. The proposed system should work on multiple devices, such as computers and mobile devices with different operating systems. Web application is one of the easiest ways to implement multiplatform and –device support. Web application means in this context a software that runs in browser. When evaluating the possible applications and solutions, web application is preferable to desktop application. Web application has the advantage of easier maintenance and install process than desktop application.

The proposed system should be easily scalable. Main motivation for this thesis is to research whether it is possible to implement the proposed system possible to implement. This is why these requirements are the bare minimum. In case of the best outcome user would have the graph she/he wanted in matter of minutes. At the moment the time spent

fetching the resources is not as important as getting the results right. The proposed system should provide new information, which otherwise would be hard to reach.

## 3.2 Prior Art

This chapter describes what prior art (applications, solutions or application programming interfaces) already exists. In the best scenario there would be already an application, which implements the complete proposed system. Unfortunately after several searches, no single application met the requirements and the technology conference story. However there were many applications and APIs, which took care of one or multiple important sections. These alternatives are divided into following subsections: 3.2.1 Seach engines, 3.2.2 Scraping, 3.2.3 NER and 3.2.4 Graph visualization.

### 3.2.1 Search engines

Search engines are good at finding meaningful websites for the user. Search engines at the moment are not providing the user aggregated information about the contents of the search results. The proposed system is about providing aggregated information in form of graph based on users' request. The existing search engines could be used in the proposed system, but only as help for getting the right resources for filtering.

Today probably the most popular search engine is Google. Other commonly known search engines are Microsoft Bing, Yahoo Search and DuckDuckGo. Almost all current search engines are web applications. Figure 9 presents the US search share change search share from November 2014 to February 2015. As can be seen in Figure 13 Google has the indisputable first place in search engines. Bing and Yahoo are competing against each other, they are only in 3 % margin from each other.

Although DuckDuckGo is much smaller search engine than the other three, it has gained some attraction due to users privacy concerns. At least Firefox and Safari browsers have an option to choose DuckDuckGo as the default search engine for the browser. This means it worth taking into account. Unlike the three other search engines mentioned before, DuckDuckGo will not collect or store any user data. It was founded back in 2008. DuckDuckGo concentrates on instant answers in addition to privacy. DuckDuckGo was only one of those four search engines not providing public search API. The reason behind this decision was that they do not have the rights to fully syndicate their search results. Although DuckDuckGo does not provide public search API, it has free instant answer API. This API provides instant answers like topic summaries, categories or disambiguation. Those features might be useful for the named-entity recognition in the proposed system. ("DuckDuckGo" n.d.)

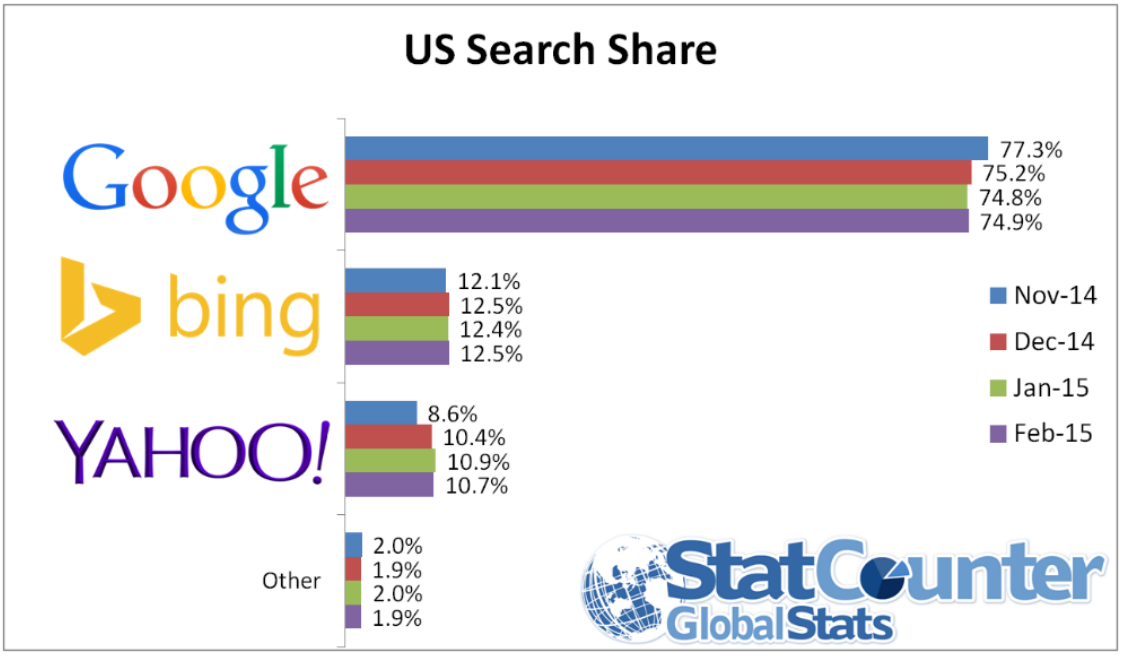


***Figure 13.*** *US Search Share February 2015.* ("StatCounter Global Stats" n.d.)

Yahoo's search engine is the oldest of all four search engines mentioned above. Its search results were first powered by Inktomi and later Google. Not until 2004 Yahoo launched its own search engine technology.("Yahoo: An 18-Year Timeline of Events | PerformanceIN" n.d.) In 2009 Microsoft and Yahoo announced a ten year deal that Microsoft Bing search engine would start to power Yahoo's search ("Microsoft and Yahoo Seal Web Deal" n.d.).

Microsoft Bing has also long history and many names. The search engine was first published to people in 1998 as "MNS Search". Back then MSN Search did not use its own search results ("MSN Search Bot a Glimpse of Ambitions" n.d.). Microsoft also developed its own search engine and started to use it in late 2004. Microsoft Bing was born in 2009, when Microsoft rebranded their search engine ("Meet Bing, Microsoft's New Search Engine" n.d.).

Google's history was discussed lengthily in the previous chapter. Microsoft Bing, Google and Yahoo search engines have public search API. Figure 14 presents the prices of these APIs. Bing and Yahoo still have separate public search APIs, although Bing provides the search engine for Yahoo. Microsoft Bing search engine is the only one to offer free tier for their search API. User can make 5000 queries per month for free by using Microsoft Bing. 5000 queries results to 250 000 results in total, which is quite a lot. Googles custom search API was by far the most expensive for all query quantities. Surprisingly Yahoo is the cheapest option when amount of queries exceed 10 000 queries.

| Queries/Results (50 results returned per query) | Microsoft Bing ("Bing Search API" n.d.) | Google ("Custom Search JSON/Atom API" n.d.) | Yahoo ("Yahoo BOSS – Pricing" n.d.) |
|---|---|---|---|
| 1000 queries per month, 50 000 results total | $0.00/mo. | $25.00/mo. | $1.80/mo. |
| 5000 queries per month, 250 000 results total | $0.00/mo. | $125.00/mo. | $9.00/mo. |
| 10 000 queries per month, 500 000 results total | $20.00/mo. | $200.00/mo. | $18.00/mo. |
| 20 000 queries per month, 1000 000 results total | $40.00/mo. | $400.00/mo. | $36.00/mo. |

***Figure 14.*** *Search API price comparison.*

Search API is good option for the proposed system. These search engines know their business and provide good quality results. Search is anyway necessary to reduce the amount of websites to crawl. In the technology conference case getting the starting websites through search engine is easier than crawling through huge amount of sites trying to find any references to conferences or to certain technologies.

### 3.2.2 Scraping

Scraping is the only way to get the data from websites, though it is possible to visit every website by hand and save the source code to a file. Collecting the webpages by hand is not efficient and would take long time. There are multiple options for scraping the web programmatically. One of the most popular solutions is to program your own scraper using already existing frameworks. Fortunately there are also solutions where user does not need to know how to program or set up a server.

Import.io ("Import.io" n.d.) is one of example of a scraping solution, which does not require programming. The service is based on visual scraping. User highlights and selects the wanted data from the web page. Import.io will crawl the selected web page and provide an API to the information selected by the user. Import.io is a desktop application, which works on Windows, Mac and Linux. The application has browser built in, which is used to select the wanted data. After the selection of wanted data is done, API is ready to be used in applications. Other alternative for visual scraping is Kimono Labs ("Kimono" n.d.), which does not require desktop application. It works as a plug-in for

Google Chrome browser. Basically it otherwise works as the Import.io. User navigates to a web page, selects the information to crawl and gets an API to that data.

In the case of frameworks, they need also custom implementation (a program) to give them a list of URLs to crawl. For the proposed system the visual scraping systems are too slow and require too much resources. Crawling should be automated in the proposed system. Doing crawling programmatically beats visual crawling in efficiency and resource usage. Although visual scraping probably has the higher accuracy, because user chooses the data not some algorithm.

### 3.2.3 NER

Named-entity recognition is essential for the proposed system to recognize the entities from the webpages. Natural Language Processing (NLP) is also needed to recognize if website is the right genre. Machine learning technics and current technology have enabled wide offering of NLP and NER services. There are multiple services that provide NER and NLP as a service through an API. In addition those can also be implemented through frameworks. Figure 15 displays three open source NER frameworks, which also include named-entity recognition. All three frameworks use statistical models to identify people, location and organizations. These statistical models can use all three techniques: word-level, list-look up and document features presented in previous chapter. Depending on the framework user can choose which models to use or just download all existing models.

| | Stanford NER | Apache OpenNLP | NLTK |
|---|---|---|---|
| License | GNU General Public License v.3 | Apache License Version 2.0 | Apache License Version 2.0 |
| Programming language | Java | Java | Python |

***Figure 15.*** *Open Source NER Frameworks.*

One of the most known NER frameworks is open source framework Stanford NER(The Stanford Natural Language Processing Group, 2015). It is licensed under GNU General Public License (The GNU General Public License v3.0, 2015). The framework recognizes places, people and organizations better than other named-entities. There are also other models available, but those three are the most comprehensive. Like the name already states Stanford NER is developed by The Natural Language Processing Group at Stanford University. The group has also developed Stanford CoreNLP, which also includes NER capabilities. The framework is written in Java. Though it has multiple different wrappers written in different programming languages.

Another alternative for open source NLP framework is Apache's OpenNLP (Apache OpenNLP, 2015), which is written in Java too. OpenNLP is licensed under Apache License (Licenses, 2015). The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It also includes NER capabilities, but NER has to be used through OpenNLP. There is no separate software for NER, like the Stanford group had. OpenNLP NER uses models to identify named-entities. Possible models at the moment are date, location, money, organization, percentage, person and time name finder.

In addition to these two, there is NLTK framework (Natural Language Toolkit, 2015), which is written in Python. It has been licensed under Apache License (Licenses, 2015). The only main difference to frameworks mentioned earlier is the programming language. NER framework needs a program to run the framework. In case of the proposed system this would mean separate NER program. The separate NER program would be a backend service providing an API to identify named-entities from text. Other alternative is to use wrapper to bridge java or python to PHP or Node.js. Every framework mentioned has wrappers for multiple programming languages. Disadvantage of wrappers is complexity.

NER APIs are easier to use than the wrappers, because there is no need to make a bridge between programming languages. Those NER APIs take a text, URL or HTML file and return intended result. For example if user wanted to recognize entities from a text, NER API response would be a list of entities found in the text. Figure 16 presents five different APIs, which offer NER as an API. Those five APIs are AlchemyAPI ("AlchemyAPI " n.d.), Open Calais ("Thomson Reuters | Open Calais" n.d.), Semantria ("Semantria | API" n.d.), Saplo ("Text Analytics from Saplo" n.d.) and TextRazor ("TextRazor" n.d.). Figure X compares features of these APIs.

AlchemyAPI and TextRazor are only APIs, which accept URL as input format. In fact only AlchemyAPI and Open Calais accept also HTML as input format. Ability to send URL instead of text to API would remove the need to scrape the web page. Though in this case, the quality of entity list takes on bigger role, because original source code is not processed. All five NER APIs have REST API and return the response at least in JSON format. REST is an acronym of Representational State Transfer. REST is a coordinated set of architectural constraints that attempts to minimize latency and network communication while at the same time maximizing the independence and scalability of component implementations (Fielding and Taylor 2000). JSON is a text format that facilitates structured data interchange between all programming languages (Ecma International 2013). JSON format enables use of REST without opinion about how the API should be used.

All five NER APIs provide also a free-tier, which enables the use of their API without payment. Sematria was the only one of the NER API providers to offer only bulk amount of free requests (maximum of 20 000 requests) regardless of the timeframe. After all free requests have been used, user has to update to paid account in order to continue use of

Semantria. Although 20 000 request would be enough for the prototype of the proposed system, the bulk system is not optimal for the proposed system.

| | Input format | Free requests per month | Terms of Use / Service |
|---|---|---|---|
| AlchemyAPI | Text, URL, HTML | 30 000 | Display AlchemyAPI logo and provide clickable link to their home page |
| Open Calais | Text, HTML | 150 000 | Display the Open Calais icon logo and link the logo to their home page |
| Semantria | Text | 20 000 ( total of free requests) | Provide attribution to Semantria ( logo, "powered by Semantria ) |
| Saplo | Text | 2 000 | - |
| TextRazor | Text, URL (beta) | 15 000 | - |

***Figure 16.*** *NER APIs features.*

Saplo is newest contender in NER and NLP API competition. Saplo was founded in 2008 in Sweden. It has also support for Swedish, which the other APIs do not have. Saplo offers 2000 request to their API per month. It is relatively small number when other competitors offer at least 15 000 requests per month. The proposed system should support only English. TextRazor offers 15 000 request per month. It is way more than Saplo but still less than AlchemyAPI and Open Calais. However TextRazor supports URL as input format, which Open Calais does not support. 15 000 requests per month is probably enough for the proposed system in this prototype phase.

Open Calais offers the most free requests per month (150 000), though there is 5000 per day request limit. Open Calais service is a part of Thompson Reuters, which is the world's leading source of intelligent information for businesses and professionals. The API also recognizes relationships, facts, events and topics. AlchemyAPI provides 1000 request per day, which equals on average 30 000 request per month. AlchemyAPI was bought by IBM in March 2015.

All three NER APIs (TextRazor, Open Calais and AlchemyAPI) are good candidates for the proposed system. It is hard to see the difference between them based only on the home pages and Figure 10. The three NER API providers were compared in NER accuracy to give some insights of their strengths and weaknesses. Text was chosen as input format, because it was only input format all three NER APIs supported. Test included two texts

from JSConf US 2015 web page ("JSConf US 2015 - The Best Conference for JS and the Web. Period" n.d.). This example was chosen, because it also brings the example case to life. One text was about sponsors of the conference and another one was about speakers of the conference. Text was extracted from the webpages. Text was only from inside of HTML body-tag, document header was discarded. Request to identify named-entities in sponsor text and speaker text was send to each of the three NER APIs. Every NER API responded with JSON, which contained found entities. Figure 17, Figure 18 and Figure 19 present the different JSON objects for same PayPal company entity. Figure 17 presents what information AlchemyAPI sends about the named-entity it has recognized.


```
{

        "type": "Company",
        "relevance": "0.503731",
        "count": "2",
        "text": "PayPal",
        "disambiguated": {
                "subType": [ "VentureFundedCompany"],
                "name": "PayPal",
                "dbpedia": "http://dbpedia.org/resource/PayPal",
                "freebase": "http://rdf.freebase.com/ns/m.01btsf",
                "yago": "http://yago-knowledge.org/resource/PayPal",
                "crunchbase": "http://www.crunchbase.com/company/paypal"
        }
}
```


***Figure 17.*** *PayPal entity information from AlchemyAPI.*

AlchemyAPI had the shortest response objects of the three NER API providers. The response tells basic information and gives URLs to get more information about the entity. AlchemyAPI has lot of external sources of information, Figure 17 shows already four different sources DBpedia, Freebase, Vago and Cruncbase. Figure 18 presents what information Open Calais sends about the named-entity it has recognized.

Open Calais presents much more information in their response. Instance-property expresses every mention found by Open Calais. Confidence scoring in confidence-property indicates the probability that the extracted e.g. person or company is indeed a person or company The NER API returns also relations with the response. Open Calais relies only to its own named-entity database to afford extra information about the entities. In the response resolutions is the link to extra information about PayPal. Figure 19 presents what information TextRazor sends about the named-entity it has recognized.

Unlike Open Calais TextRazor provides extra information about named-entities through Freebase, Wikipedia and Wikidata. TextRazors assigns type of named-entity differently than the two previous NER APIs. For example type of PayPal is classified as agent, company and organization in Figure 19. In addition TextRazor entity information also presents all freebase types the entity is part of.

```
{
        "_typeGroup": "entities",
        "_type": "Company",
        "forenduserdisplay": "false",
        "name": "PayPal",
        "nationality": "N/A",
        "confidencelevel": "0.841",
        "_typeReference": "http://s.opencalais.com/1/type/em/e/Company",
        "instances": […],
        "relevance": 0.2,
        "resolutions": [
            {
                "permid": "4295902034",
                "score": 0.4241735,
                "name": "Paypal Inc",
                "commonname": "Paypal",
                "id": "https://permid.org/1-4295902034"
            }
        ],
        "confidence": {
            "statisticalfeature": "0.905",
            "dblookup": "0.0",
            "resolution": "0.4241735",
            "aggregate": "0.841"
        }
}
```


***Figure 18.*** *PayPal entity information by Open Calais.*


```
{
       "id": 254,
       "type": [
              "Agent",
              "Organisation",
              "Company"
       ],
       "matchingTokens": [ 1488 ],
       "entityId": "PayPal",
       "freebaseTypes": [
              "/Internet/website_owner",
              "/book/book_subject",
              "/business/business_operation",
              "/organization/organization",
              "/organization/organization_partnership",
              "/finance/currency",
              "/venture_capital/venture_funded_company",
              "/business/employer"
       ],
       "confidenceScore": 7.27548,
       "wikiLink": "http://en.wikipedia.org/wiki/PayPal",
       "matchedText": "PayPal",
       "freebaseId": "/m/01btsf",
       "relevanceScore": 0.444943,
       "entityEnglishId": "PayPal",
       "startingPos": 8690,
       "endingPos": 8696,
       "wikidataId": "Q483959"
}
```


***Figure 19.*** *PayPal entity information by TextRazor.*

Sponsor text from JSConf US website had total of 27 sponsors listed. Those sponsors supported the conference. Speaker text from JSConf US website had total of 40 speakers listed. Those speakers are persons, who talked at the conference. Number of sponsors and speakers were calculated manually straight from the websites. Occurrences of correctly classified speakers and sponsors from the responses was also calculated by hand, because

sample size was so small. High accuracy was also a partial reason why occurrence calculation was made by hand. Figure 20 presents how accurately each NER API recognized companies and persons from sponsor and speaker text.

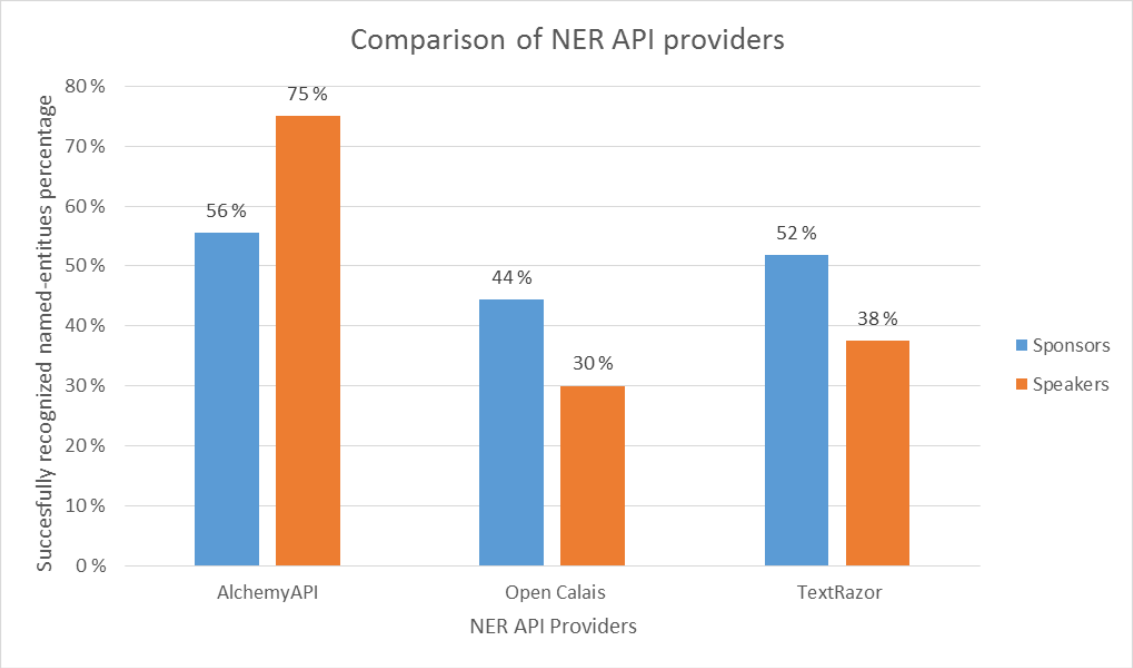


***Figure 20.*** *Comparison of NER API accuracy.*

AlchemyAPI recognized 15 companies from the total of 27 companies and 30 people from the total of 40 people as named-entities. In other words, AlchemyAPI recognized a bit over half (56%) of the sponsors and 75% of speakers. Open Calais recognized 12 companies, which is 44% of companies listed. It recognized only 12 people, which is 30% of people listed. TextRazor outperformed Open Calais but did not reach AlchemyAPIs level. TextRazor recognized 52% of companies listed in sponsor text and 38% of people listed in speaker text.

Difference in sponsor text named-entity reorganization was not remarkable. AlchemyAPI recognized 15, Open Calais 12 and Text Razor 14 sponsors. Each of those NER APIs could be used with the proposed system, because there were only two sample texts and difference between them was minimal. Though AlchemyAPI beat the other two NER APIs in people recognition with twofold accuracy performance percentage. The small study between the NER APIs indicated the challenge to identify named-entities from text. Probably the accuracy would be better if input format would be HTML and also the structure would be part of NER. AlchemyAPI recognized one company more, when input format was HTML straight from the sites source. Use of only NER API will not be the best solution for proposed system. Probably the best solution is to combine those APIs and also own NER techniques based on the structure.

### 3.2.4 Graph visualization

One of the key points of the proposed system is the visualization. Without the visualization it would only be information extraction system. There are desktop and web applications, which visualize graph data. The applications support importing the graph as a file. For example Gephi ("Gephi - The Open Graph Viz Platform" n.d.) is an open source graph visualizing software for desktop computers. It is an interactive visualization and exploration platform for all kind of networks and graphs. Figure 21 presents how graphs can be visualized in Gephi. Visualizing the results of search for the proposed system could be done with Gephi.

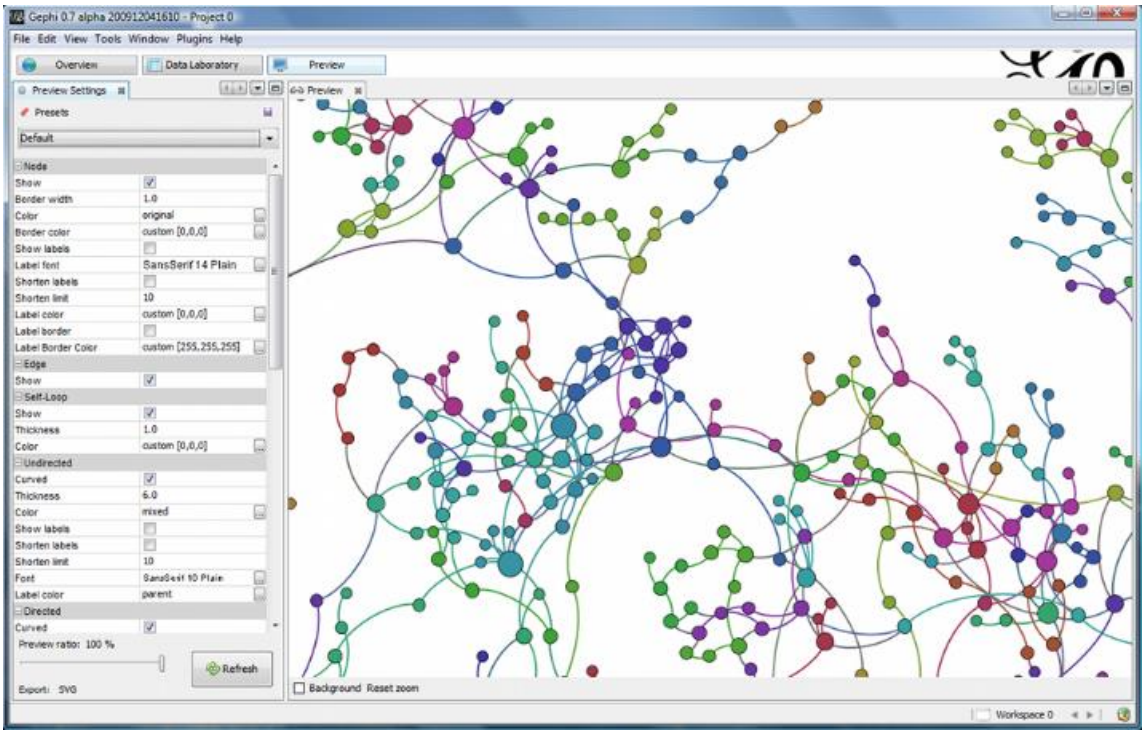


***Figure 21.*** *Screenshot from Gephi application.*

However visualizing the data with Gephi would not serve the goal of easy to use system. The user is then forced to first get the graph information through the proposed system and import that information to Gephi. The software has an advantage of providing all necessary tools to shape the graphs appearance as seen in Figure 21. Linkurious provides a simple web interface to visualize graphs. User has to import graph data via file or Neo4j-database or create a new graph. It is free for open source projects but otherwise it is subject to charge (990€ per year per seat / enterprise 1990€ per year per seat). Linkurious advantage is its web interface, because user can use the web interface from anywhere without any installation hassle. These applications are worth considering if the proposed system would be divided into pieces.

There are also multiple graph drawing libraries available, especially for JavaScript and web. Figure 22 presents four different JavaScript graph drawing libraries and their features and licenses. Rendering in these libraries has been done using one of the three technologies SVG (Scalable Vector Graphics), HTML5 canvas element and WebGL. W3Cs definition for SVG is "a markup language for describing two-dimensional graphics applications and images, and a set of related graphics script interfaces" ("W3C SVG Working Group" n.d.). "The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, art, or other visual images on the fly" ("4.11 Scripting — HTML5" n.d.). WebGL is a cross-platform, royalty-free web standard for a low-level 3D graphics API based on OpenGL ES 2.0, exposed through the HTML5 Canvas element as Document Object Model interfaces("WebGL - OpenGL ES 2.0 for the Web" n.d.).

| | D3.js | Sigma.js | Linkurious.js | Cytoscape.js |
|---|---|---|---|---|
| License | BSD License ("The BSD 3-Clause License " n.d.) | MIT License ("Sigma.js MIT License" n.d.) | GNU General Public License v.3 ("GNU General Public License v.3" n.d.) / Commercial License | LGPL License ("Cytoscape.js LGPL-License" n.d.) |
| Rendering | SVG | WebGL/Canvas | WebGL / Canvas | Canvas |

***Figure 22.*** *Visualization library comparison.*

D3.js library ("D3.js - Data-Driven Documents" n.d.) is the most general purpose library of the four libraries mentioned in the Figure 22. D3.js is JavaScript library for manipulating documents according to data. D3.js uses HTML and SVG to render the figures and graphs. In addition of being the most general library D3.js is also not as high-level library as the three others are. D3.js library is good alternative for visualizing the graph of the proposed system. Sigma.js is a JavaScript library dedicated to graph drawing ("Sigma Js" n.d.). It uses WebGL to render graphs. If the browser does not support WebGL, Sigma.js renders graphs with canvas. It is licensed under MIT License, which basically gives the right to use, modify and distribute the code however user wants.

Linkurious.js is based on the web application Linkurious mentioned before ("Linkurious.js Graph Visualization Library" n.d.). In fact it is based on the Sigma.js library. Linkurious.js basically extends the Sigma.js with more high level features, such as styling in the UI, camera animation, file and image exports. Linkurious.js is also the only one with commercial license in addition to GPL v3 license. The commercial licenses available are 30 000 € per year for corporate for one commercial project with unlimited

developers. GNU General Public License v3 is free and has a strong copyleft software license.

The last library in Figure 22 is Cytoscape.js, which renders the graphs only on canvas ("Cytoscape.js" n.d.). The library is LGPL Licensed, so it is less permissive than the GPL. Cytoscape.js defines itself as a graph theory library for analysis and visualization.

From these four libraries Sigma.js and Cytoscape.js are the best candidates for the proposed system, because the proposed system will focus on displaying the information with graphs. However Neo4j itself uses D3.js to show graphs in their web admin interface. D3.js is trusted by the graph database itself, it is a good reason to still consider it. Sigma.js and Cytoscape.js also have least intrusive licenses because MIT and LGPL License will not bind the proposed system like the GLP would. Linkurious.js would also be among the best candidates for the proposed system, if it were not for GPL License. MIT License is by far the best license to use in the proposed system, because it has the least restrictions for the use of the software licensed. However the license will not be a problem for the proposed system, because its main purpose is to test if the idea works at all.

## 3.3 Proposed solution

Above, we introduced already existing search engines and their public APIs. Proposed solution for the search part would be designed and implemented by me. As discussed in previous chapter each of the search engines has been running for at least more than five years and have their own algorithms in place to rank the results. Implementation of my search engine for the proposed system would require a lot of effort and time. Still it might not be anywhere near in precision and usefulness as the current search engines. The proposed system should be as general as possible. This eliminates the possibility to build search engine only for certain area like technology conferences. Existing search engines are the best choice for the proposed system.

As already discussed in section 3.2.2, automatic scraper makes more sense than visual scraper. There are multiple scraping frameworks and libraries available for multiple programming languages. The proposed solution for scraping would take advantage of existing libraries and frameworks. A crawler to crawl wanted sites is straightforward to implement. One of advantages in implementing own scraper is the possibility to build own algorithm to choose what is the relevant data from the scraped website. The proposed system is just a prototype, so disk space for the scraper does not need to be huge and efficiency does not need to be high.

NER API is better solution for the proposed system than the frameworks, because those frameworks need language bridging from one programming language to another programming language. In addition NER APIs have dynamic database of named-entities, while other frameworks only work with data imported to the system. For the prototype NER API gives more flexibility. In short, own implementation of NER would take a lot of effort and time. It is not worth the effort and time, because there are many alternatives already available like the frameworks and NER APIs presented in subsection 3.2.3. Probably the best solution would be to use NER API and combine it with own algorithm, which also takes structure of HTML into account. If user provides information about the structure of wanted information, then use of HTML structure in NER should be simpler.

Subsection 3.2.4 introduced the existing alternatives to visualize graph data. Desktop application for visualization is easy find, deploy and use, but it would break the user experience. The proposed system should return the data in some standard format through API to the web application regardless of the visualization solution. The proposed system should only provide means to observe the information, not edit it. Sigma.js, Cytoscape.js and D3.js are all comprehensive and extensive libraries for graph visualization. It would be fool's errand to design and implement another graph visualization library. One of these libraries will be great solution for the proposed system to visualize the graphs.

## 3.4 Summary

The proposed system will include search, scraping, named entity tagging and visualization of the information. Combination of different applications (for example desktop and web) does not provide seamless user experience. The proposed system will be implemented as own solution, which combines different APIs and own code. As already previously discussed search API is best solution to filter out unnecessary websites. The search APIs under consideration were Bing, Google and Yahoo search APIs. Bing offers highest amount of free queries per month (5000 queries per month). Bing is good choice as search API of the proposed system.

Scraping solutions presented in previously were visual and automatic scraping. Automatic scraping is done programmatically. In visual scraping each website has to be visited in order to choose what data should be scraped. Automatic scraping needs just a list of seed websites, were to begin its crawling process. For the proposed system automatic scraping is better choice, because it does not need human intervention to scrape intended websites. Automatic scraping for the proposed system will be combination of existing libraries or frameworks and own code.

Different ways to recognize named-entities was discussed in previous sections. These included open source frameworks and commercial NER APIs. Frameworks require too much effort to make them usable for the proposed system. NER API is better solution for the proposed system, because it is very simple to implement into whatever program and

information is up-to-date. Visualization is easiest to implement with help of visualization libraries. As discussed earlier, there are good options for visualization library like Cytoscape.js or Sigma.js. Sigma.js has more permissive license, though Cytoscape.js provides more UI actions and elements. Sigma.js will be used for the prototype of proposed system, because of better license and WebGL support. Next chapter describes my implementation of the proposed system, its architecture and technologies used in it.

# 4. IMPLEMENTATION

As discussed above, there are multiple options for the implementation of the prototype application. Chapter 3 proposes a combination of own implementation and multiple APIs. This chapter concludes the structure and technologies used in the prototype application.

The prototype application uses Model-View-Controller (MVC) pattern. This means the application is divided into three interconnected parts: model, view and controller. The MCV pattern is visualized in Figure 23. Model is the place, where all data is saved and handled. View stands for the user interface in the prototype application. Controller handles the application logic in the prototype application. In the prototype application the view is on client-side and model and controller are on server-side. The client and server communicate through a representational state transfer (REST) application interface. Benefit of the MVC pattern is the clear division of application logic, user interface and data handling.

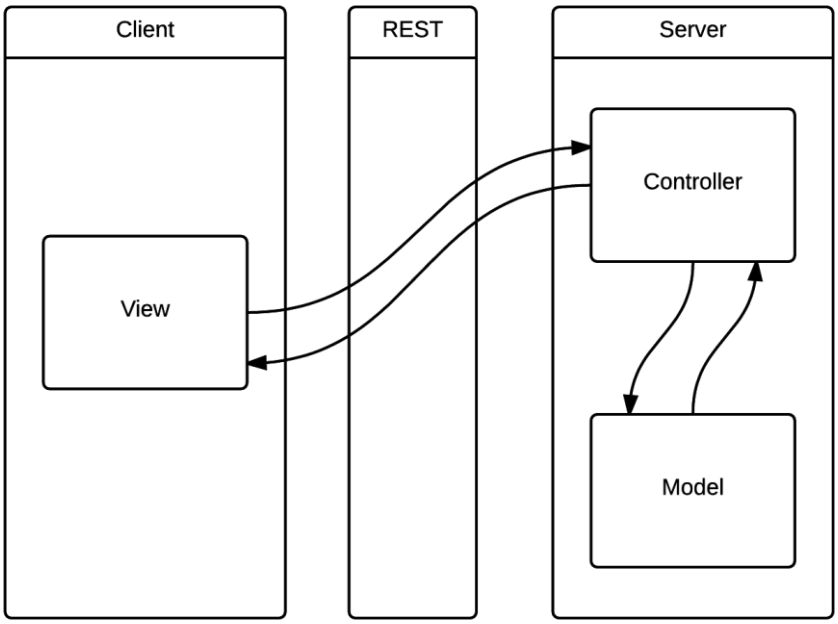


***Figure 23.*** *MVC pattern of the prototype application.*

The prototype application will be a web application, because of the versatility web application. There is no need for different desktop application, because almost every computer has browser and also tablets have usually browser. This chapter is divided into sections according to the MVC pattern. Section 4.1 explains the model part of the prototype application. Section 4.2 presents the view part of the prototype application. Section 4.3 discusses the controller part of the prototype application.

## 4.1 Model

The prototype application uses two different NoSQL databases to save the data gathered from web sites. The reason why two databases used will be further discussed in section 4.3. The “No” in NoSQL can mean literally there is no SQL used in a database, or it can mean “not just SQL”. SQL is acronym for Structured Query Language. SQL is the query language traditionally used with relational databases. NoSQL databases were developed to address limitations of relational databases. These limitations appeared when the number of data and users in web exploded. Number of web companies like Google, Facebook and LinkedIn needed to support high availability, low latency response times and large volumes of read and write operations. Relational databases were not able to support these. (Sullivan 2015)

NoSQL databases are well equipped to handle unstructured data and give user flexibility on database schema design. Unlike NoSQL relational databased are at their best when those are used to save structured data. Scalability is also an advantage in NoSQL, because most of the NoSQL databases can be distributed over multiple computer instead of only one. (Leavitt 2010) NoSQL databases can be divided into four major types: key-value databases, document databases, column family databases and graph databases.

MongoDB is document database, which means collection of data items stored together in a flexible structure. “A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.” (“Introduction to MongoDB — MongoDB Manual 3.0” n.d.) MongoDB was chosen to save all the data incoming from different APIs, like Bing and AlchemyAPI and scraped HTML. MongoDB was chosen because of the flexible schema and JSON compatibility. Every API used in this prototype application sends their response as JSON array.

Neo4j is a graph database. Graph databases are part of the NoSQL movement explained already earlier. Instead of saving data to tables, in graph database the data is saved as nodes and arcs between the nodes. This is a great advantage when the data to save has many connections between each other. In case of traditional relational database relationships are saved as rows to linking table. The goal for the prototype application is to show the connections between entities as a graph. This is why Neo4j was chosen as the database to save entities found from crawled websites. Figure 24 describes how the data of prototype application was divided between these two databases. Basically only the final result graph data was saved to Neo4j and everything else was saved to MongoDB.

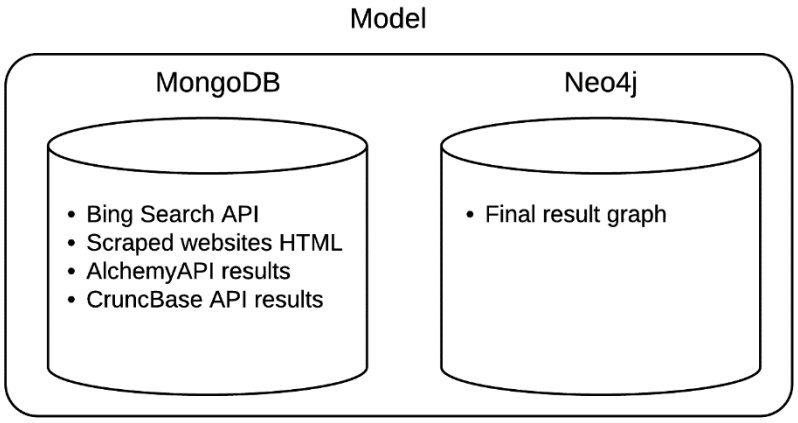


***Figure 24.*** *Databases used in prototype application and data saved to those.*

Neo4j does not use SQL as its query language, like name NoSQL refers. Instead of SQL Ne4j graph database uses Cypher Query Language, which is declarative graph language. Cypher is designed to be a humane query language suitable for developers and operations professionals, and hence, elegantly combines simplicity, expressiveness and efficiency ("8.1. What Is Cypher? - - The Neo4j Manual v2.2.6" n.d.). Figure 25 demonstrates the syntax of the Cypher Query Language.

```
SQL Query:
SELECT *
FROM Company, SponsorRelationship, Event
WHERE Company.id = SponsorRelationship.companyid AND Event.id = Sponsor-
Relationship.eventid = Event.id AND Event.name = 'JSConf 2015';

Cypher Query:
MATCH(company:Company)-[relationship:SPONSORS]->(event:Event)
WHERE event.name = 'JSConf 2015'
RETURN company, relationship, event
```

***Figure 25.*** *Example of same query in SQL and Cypher Query Language.*

Like the Figure 25 shows Cypher query is simpler and more declarative than the SQL query in this example case. Neo4j also provides a graph view to its graph database through their database management web application. Figure 26 shows an example of Neo4j graph view. The result shown in Figure 26 corresponds the query made in Figure 25. The data used here is shown in the Figure 26 is from the tests, which were run on the prototype application.


***Figure 26.*** *Example of Neo4j graph view.*

## 4.2 View

This section describes the technologies used in the view of the prototype application. These technologies include the programming language JavaScript, HTML and CSS. HTML is acronym for Hypertext Markup Language. HTML is the predominant markup language used to describe content, or data, on the World Wide Web (another lesser-used markup language is XML) (Alexis Goldstein, Louis Lazaris 2015). HTML is considered to be a markup language because it tells the browser how to structure the content on the website. HTML5 is the newest major version of HTML. CSS stands for Cascading Style Sheet. CSS is a style language that describes how HTML markup is presented to the user (Alexis Goldstein, Louis Lazaris 2015) . The combination of JavaScript, HTML and CSS is called HTML5 and related technologies. Often the combination of all three technologies mentioned before is just referenced as HTML5 without reference to related technologies, although CSS and JavaScript are not a part of the HTML5 specification.

JavaScript is a programming language most known from its extensive use in browsers to make web applications. JavaScript was introduced in 1995 as a way to add programs to web pages in the Netscape Navigator browser. The language has since been adopted by all other major graphical web browsers. It has made modern web applications possible - applications with which you can interact directly, without doing a page reload for every action. JavaScript is used in the prototype application to enable interactions and graph drawing in the browser. (Haverbeke 2014)

As already mentioned JavaScript was developed as a programming language for browsers, which is why it is used at client side as an interpreted language. Although most of modern browsers in 2015 use compilation to compile JavaScript to machine code. ECMAScript standard was developed to determine how JavaScript should work, what features it has and enable and ensure its compatibility cross different browsers. ECMAScript 2015 is the latest ratification of ECMAScript, which introduces lot of new features to JavaScript, such as classes. These new features were not used in the prototype application. Subsection 3.2.3 presented multiple alternatives for graph drawing. D3.js was chosen as the library to draw graphs for the prototype application, because Neo4j also uses it to provide the graph view to its database.

User has a clear picture in his/her mind what information the application should gather and show as graph. User navigates to the landing page of prototype application and writes down to the form what should be searched and how the data is linked. User wants to see technology conferences, which are focused on JavaScript, Node.js and HTML5. Connections between the speakers of the conference and sponsors of the conference are interesting, so user wants to visualize also those in the graph. User request has to be written down as shown in Figure 27.

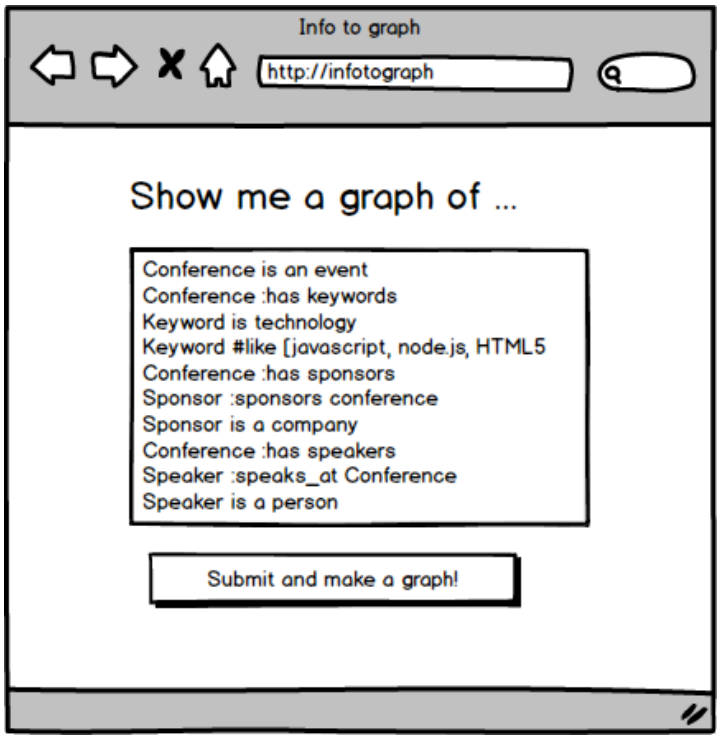


***Figure 27.*** *User Interface for users request.*

The first thing to define should always be the entity from which the other information should be gathered. As in the technology conference case, the first thing to define is the conference, because all the following information should be found from conference webpages or related webpages. Keywords are defined next to limit the conferences to be displayed in the graph. After the main entity and keywords limiting the main entity, the related entities can be determined. Every related entity name has to be tied to the main

entity. Example of this is shown in Figure 27. The main entity has to be first defined before defining the relationship between the main entity and other entities. Relationship is defined by using character ‘:’ to identify the relationship name and a relationship between the entities. The prototype application will only recognize four different entities in addition to the main entity. After user has inserted the information and clicks the submit button. The request is sent to REST interface.

After a while user will get a response from REST interface and the browser will show the graph of the information. Example of this graph can be seen in Figure 28.

The graph shows the connection between conferences and speakers and sponsors like the user requested. By clicking the conference there is more information available.

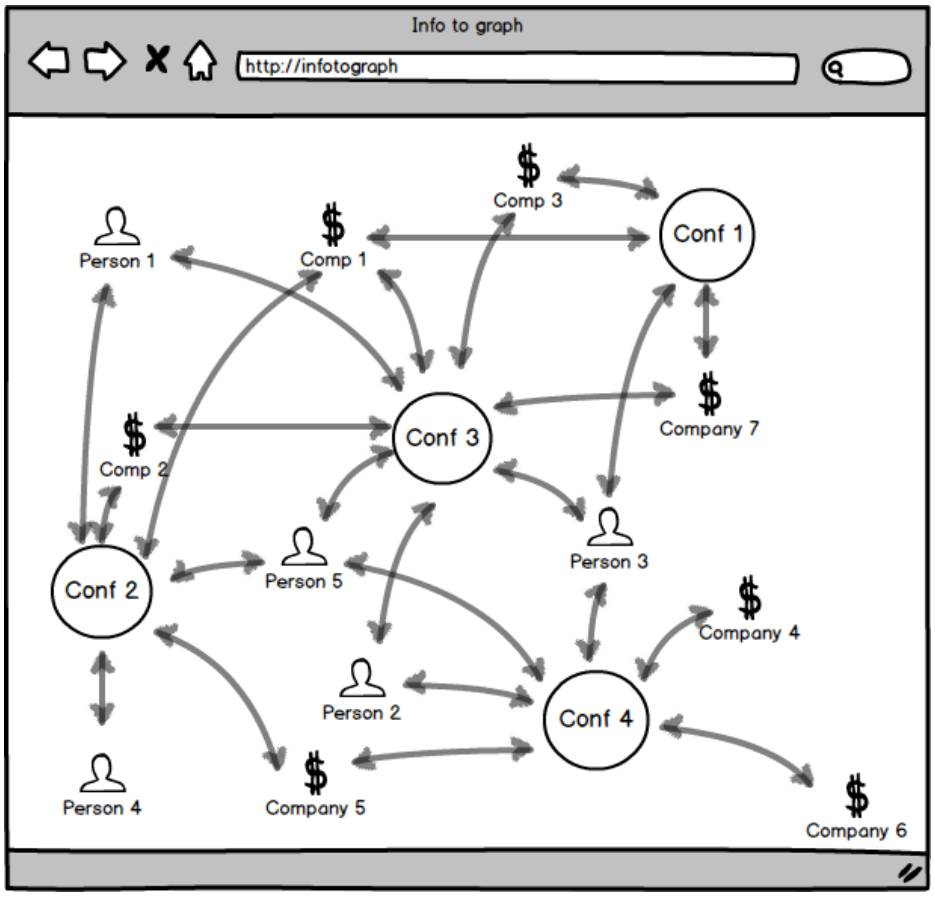


***Figure 28.*** *Example of the graph returned to user.*

## 4.3 Controller

This section presents the technologies used in server-side of the prototype application and the overall architecture of the controller. Prototype application logic, in other words the controller, uses Node.js runtime environment and as programming language JavaScript.

The use of modern compilation made also JavaScript in server side possible. Node.js is an example of this. “Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight

and efficient." ("Node.js" n.d.) Node.js applications are written in JavaScript. Node.js is commonly used to write server-side web applications. Because Node.js can used on server-side it enables the whole application to be written in JavaScript, both server- and client-side. This was the main reason why Node.js was chosen to implement the server-side of the prototype application. ExpressJS is a Node.js web application framework, which is used in the prototype application. Web application framework helps to create structure for the application. The prototype application serves the data through application programming interface in JSON format. This means the user interface gets its data JSON format by using Ajax. Ajax is acronym for asynchronous JavaScript XML. With help of Ajax web application can make asynchronous calls to retrieve data in the background without interfering the existing page.

Figure 29 shows the overall architecture of the prototype application. The UI in Figure 29 stands for user interface, which implementation was previously discussed in section 4.1 View. First the user writes the request on the web page, like shown in Figure 27. When user clicks "Submit" button, a POST request is sent through REST interface to the application logic. The entity types of the prototype application are limited to company, person, technology and event. This scope was chosen for the entities to test the prototype application without the need to identify multiple entities.

The application logic is running on a server and it has been written in Node.js. The application logic is own implementation because mixing and parsing together a coherent application from multiple software would have been wasted effort. The server side of the prototype application serves as REST interface for the UI. After the user request has been received, the application will parse and structure it to understand what it should search for and what kind of graph should be the end result. Website search is done first with the most relevant terms. In the case of the example search is done with following search terms "JavaScript conference", "Node.js conference", "HTML5 conference". These terms were chosen, because the main entity was conference and the limitations for the conference were the keywords: JavaScript, Node.js and HTML5. The search is done by using Bing Search API. It was chosen over Googles and Yahoos API, because it offered largest amount of free queries per month as presented in subsection 3.2.1.

After the search results have been received from the Bing Search API, the received results will be filtered based on ranking in search results, URL, title, and a brief description given in the search results. The filtered list of search results are crawled and then saved to database. All the crawled websites will be examined for keywords and entities like speakers and sponsors and possible links including the keywords. Those links containing keywords will be also crawled to ensure every entity is gathered.

AlchemyAPI was chosen as NER API for the prototype application because its good performance to recognize people from HTML in the test discussed in subsection 3.2.3. After all the webpages with probability of containing wanted entities are accounted for, these

webpages are sent to AlchemyAPI to recognize the entities in HTML. These entities are looked through and filtered based on the user request. For example in the technology conference case all other entities except events, people and companies are discarded. In subsection 3.2.2 proved AlchemyAPI recognizes people rather well but does not recognize companies as well as people. That is why companies are also recognized based on their URL from the HTML. The URLs are matched to CrunchBase ("CrunchBase" n.d.) dataset, which includes data of organizations including their homepages. The URL recognition was chosen in addition, because usually companies mentioned in the web sites have link to their homepage, at least in the cases they are sponsoring some event or want otherwise to promote their company.

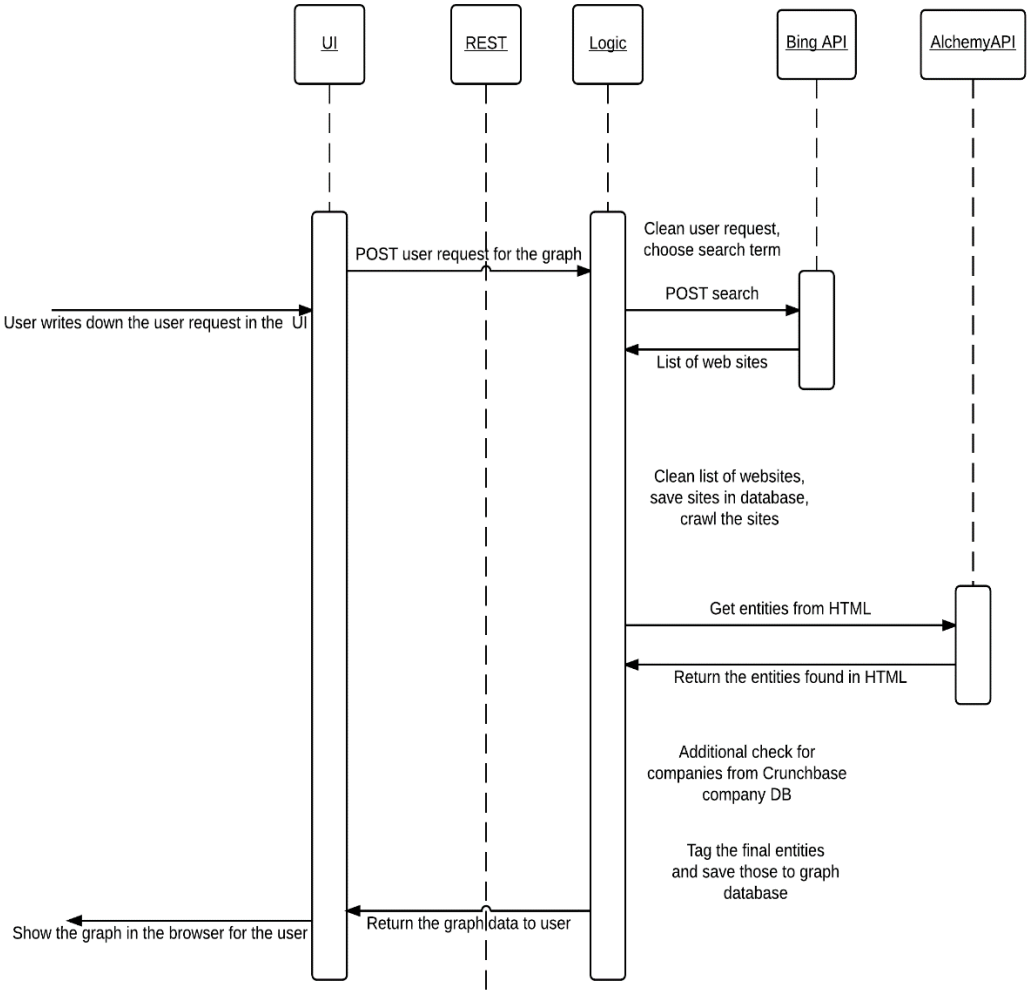


***Figure 29.*** *Sequence diagram of the prototype application.*

Found entities from the webpage are saved to graph database and linked to main entity. The main entity information is extracted from either straight from the website or through AlchemyAPIs response. After all entities and their relationships are saved to the graph database, response is sent to user to notify the graph is ready to be examined. The data from the request is saved to the graph database to enable its further use and examination.

# 5. CASE STUDY

This chapter examines the functionality and accuracy of the prototype application implemented according to previous chapter. The test case is the same as what have been used as an example through the thesis. The goal of the case study is to verify whether the prototype application succeeds to gather the right information based on users request. The case study offers a great opportunity to evaluate idea and prototype application.

This chapter has been divided into four sections. Section 5.1 presents the methods to be used in testing and also the test objectives. Section 5.2 discusses the test case and why the chosen test case was chosen. Section 5.3 presents the results of the test. Section 5.4 discussed further the results and evaluates the prototype application.

## 5.1 Methods and objectives

The case study contains two parts. Here will be description of the methods used in this case study. This section will also determine the key objectives for the case study. The case study will solve how accurately the prototype finds entities and removes those, which do not belong. The accuracy of main entity classification is important because wrong main entity information clutter the result graph. Goal of this case study is not to review how well AlchemyAPI and CrunchBase recognized entities from HTML. Rather the goal of the case study is to find out how well the prototype application found only relevant entities from websites. For example it is more important that the prototype application recognized about right amount of speakers than if AlchemyAPI did not recognize all people's names on the website. This approach was chosen because the prototype application is under review in this case study not the AlchemyAPI.

In the case study the number of correctly classified main entities is studied. The main entities are technology conferences. Correct type of the main entity is determined manually. The emphasis is on whether the prototype application has successfully classified the main entities. Precision on main entity detection ensures more informative and accurate graph to be displayed to users. In addition speakers detected from the websites should be identified as people, who spoke at the technology conference. Speakers' names should be spelled properly and match the speaker list mentioned in conference website. Sponsors detected from the conference websites should be identified as companies, which sponsored the conference. Every company name should be spelled properly and match the sponsors mentioned on the conference website.

## 5.2 Case

Scope of this case study will be limited to only one use case. Effort to manually verify the accuracy of multiple use cases is beyond this thesis. The case to be studied will be technology conferences, which concentrate on Node.js, JavaScript and HTML5. Basically it is the same as the example discussed throughout the thesis. This case will not validate the general purposes of the prototype application.

First request for the technology conference graph is asked through the user interface. The prototype application will manage this request and build graph to show how technology conferences are linked through speakers and sponsors. This user request is run only once and data collected from this user request is used to evaluate the accuracy of the prototype application. There will be five different technology conference sites picked from the result graph. All these five sample technology conferences web sites will be manually visited. The motivation is to manually identify and calculate the companies sponsoring the conference and people speaking at the conference. These manually acquired results will be compared with the results given by the prototype application in the next section.

## 5.3 Results

This chapter is divided into subsections according to the entities (technology conference, speakers and sponsors) used in the case study. Subsection 5.3.1 presents how well the prototype application recognized technology conferences. Subsection 5.3.2 discusses results of sponsor reorganization from conference sites. Subsection 5.3.3 presents how well the prototype application recognized the right speakers from conference websites.

### 5.3.1 Technology conferences

Result graph returned 62 supposed technology conferences, whose focus area is HTML5, Node.js or JavaScript. All 62 conferences with their websites have been listed in the appendix. Every one of these 62 supposed technology conferences websites were visited manually and determined whether the visited website is a technology conference homepage or not. The requirements for technology conference home page are the following: conference has speakers listed on its site or intends to have speakers at the event, conference has sponsors listed on its site or it offers possibility to sponsor the event. After manually checking every supposed technology conference site, 51 of those were confirmed

technology conferences. Figure 30 shows 82% of the supposed technology conference sites were actual technology conferences with speakers and sponsors.

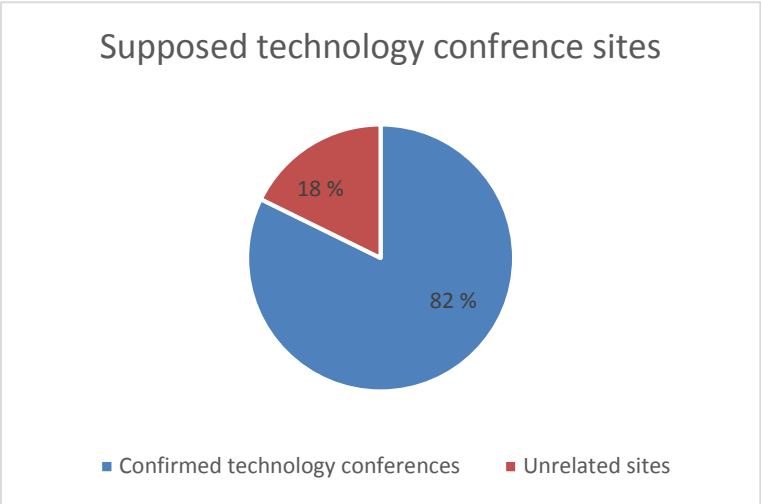


***Figure 30.*** *Distribution of confirmed technology conferences and unrelated sites.*

Unrelated sited included news sites like TechCrunch, unrelated conferences (no mention about JavaScript, Node.js or HTML5) and conference sites, which did not otherwise meet the requirements laid up previously. From these 51 confirmed technology conferences five were chosen to be examined more closely. These five conferences were ng-conf ("Ng-Conf May 4th - 6th 2016" n.d.), Nordic.js ("Nordic.js" n.d.), JSConf US 2014 ("JSConf US 2014" n.d.), Nodevember ("Nodevember" n.d.), and CascadiaFest 2015 ("CascadiaFest 2015" n.d.). Next subsection discusses the results of sponsor and speaker recognition for those five conferences mentioned previously.

### 5.3.2 Sponsors

All these five technology conference websites were manually visited. The sponsors of the technology conferences were listed into excel sheet by hand. Sponsor information from prototype application was examined as a result graph. Those results were compared and saved to the same excel sheet to ease the result comparison process. Figure 31 describes the results of the sponsor calculations. Only one of the sponsors in the result graph was not a real sponsor of the technology conferences.

| | *Conference sponsors* | *Recognized sponsors* | *Total number of sponsors in result graph* |
|---|---|---|---|
| *CascadiaFest 2015* | 17 | 11 | 11 |
| *JSConf US 2014* | 34 | 24 | 24 |
| *Ng-conf* | 0 | 0 | 0 |
| *Nodevember* | 18 | 10 | 10 |
| *Nordic.js* | 14 | 6 | 7 |

***Figure 31.*** *Results for the sponsors.*

Ng-conf did not have any sponsors listed in their website. The prototype application did not have any sponsors saved for ng-conf. Figure 32 introduces accuracy percentages for each conference. Highest percent of correctly recognized sponsors was 70 and the lowest percent was only 43. Nordic.js had the lowest accuracy in the result graph for the sponsors. The prototype application recognized only 6 of 14 sponsors mentioned at their website. The prototype application did not recognize well Nordic.js sponsors, because majority of them were Swedish and CrunchBase API has inadequate information about the Swedish companies. Sponsors of JSConf US 2014 were most correctly recognized, 70% of all its sponsors were shown in the results graph.

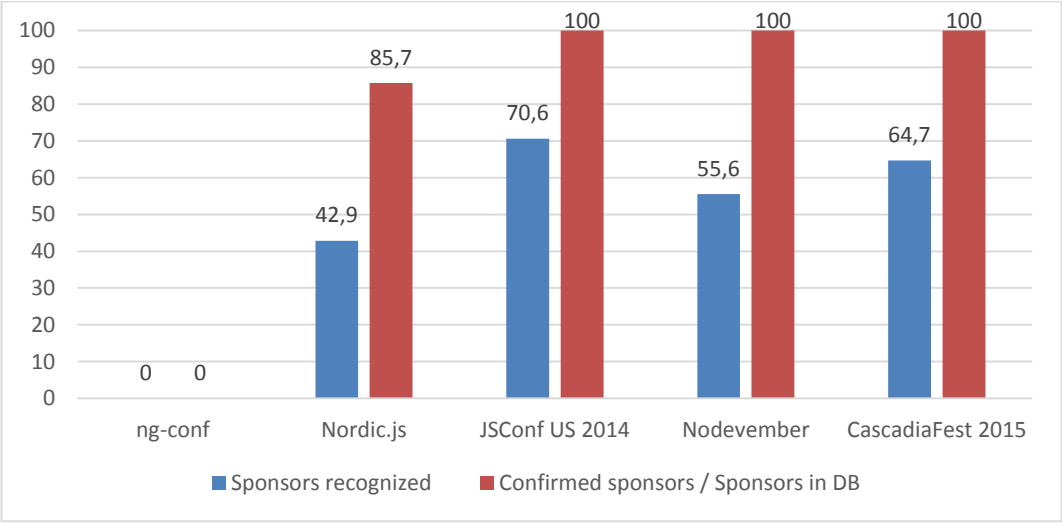


***Figure 32.*** *Prototype application results for sponsor recognition.*

### 5.3.3 Speakers

All these five technology conference websites ( ng-conf, Nordid.js, JSConf US 2014, Nodevember and CascadiaFest 2015) were visited. The speakers of the technology conferences were listed into excel sheet by hand. Speaker information from prototype application was examined as a result graph. Those results were compared and saved to the same excel sheet to ease the result comparison process. Figure 33 shows the calculated number of sponsors from the conference website and from the result graph. Overall the speakers were more precisely recognized than the sponsors.

| | *Conference speakers* | *Recognized speakers* | *Total number of speakers in result graph* |
|---|---|---|---|
| *CascadiaFest 2015* | 34 | 31 | 37 |
| *JSConf US 2014* | 26 | 16 | 44 |
| *Ng-conf* | 9 | 8 | 8 |
| *Nodevember* | 51 | 49 | 54 |
| *Nordic.js* | 14 | 11 | 17 |

***Figure 33.*** *Results for the speakers.*

Figure 34 shows the accuracy of the result graph compared to the manual calculations. The lowest percentage of recognition was 61.5 for JS Conf US 2014 and highest percentage was 96.1 for Nodevember. However there were more non-existent speakers in the result graph of the prototype application. JSConf US 2014 had the worst accuracy in speakers in the result graph. Only 36,4% of all JSConf US 2014 speakers the result graph showed were actual speakers of the conference. Rest of the false sponsors for JSConf US 2014 were malformed and combinations of the real speaker names. Luckily the prototype application had better accuracy ( over 60% ) in other conferences.

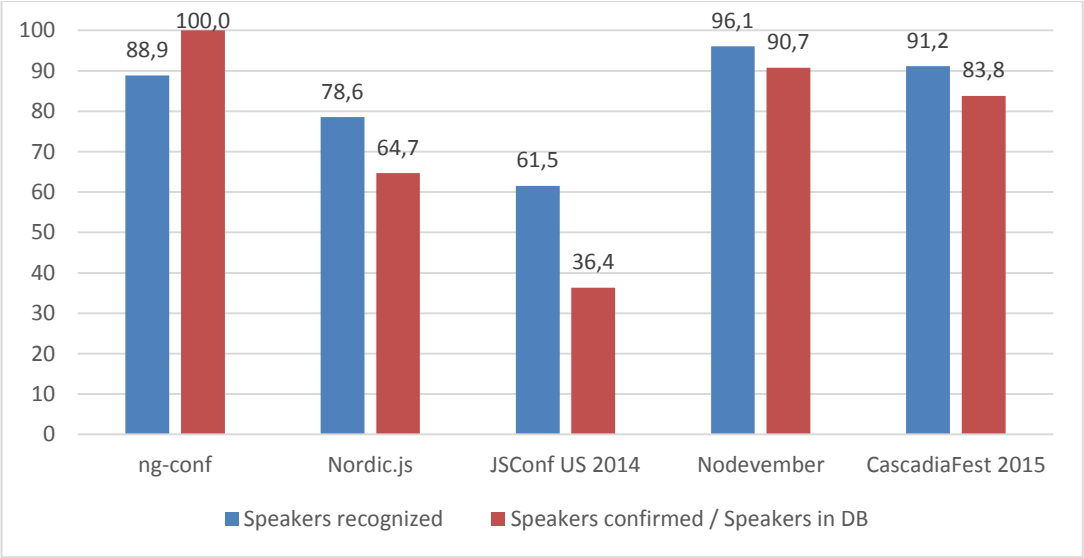


***Figure 34.*** *Prototype application results for speaker recognition.*

## 5.4 Evaluation

Based on the results presented in previous section the prototype application proved to solve the problem of visualizing linked online data. Although the sample size was small and the case study was run only once, the accuracy of the prototype application was acceptable for testing the idea. The prototype application recognized well the main entities (technology conferences), 82% of all supposed conferences were really technology conferences. The recognition of sponsors could have been better, however the sponsor data did not have so much extra noise data than the speakers had. Almost all sponsors shown in the result graph were real sponsors of the conferences. The prototype application had difficulties to recognize speakers as accurately as the sponsors. The main reason for this was that the real speaker names were combined and mixed to make new false speaker names.

The goal of the generic prototype application was not reached. The gravity of such prototype application went beyond the scope of this thesis. As said before the prototype application recognized only events, people and companies. The technologies used in the example provided a way to limit the results also in the result graph. If the prototype application would have been made from start to only recognize sponsors and speakers from technology conference websites, the results would have been more precise. The prototype application had only few precisely targeted features to improve the accuracy of sponsor and speaker recognition, which was mainly derived from the entities names given by the user. There is room for improvement and further development in the prototype application and the idea behind the prototype application.

# 6. CONCLUSIONS

The goal of this master's thesis was to study whether there already existed an application or if it was possible to create application, which would visualize online linked data according to users' request. This idea could change the way to search and visualize information of multiple named-entities and connections between them. The technology conference example was used throughout the thesis to make the idea behind the prototype application easier to understand. The ideal application would have been able to create a result graph from any user request.

This thesis researched the different possibilities to implement the idea. The alternatives were either to find existing application or do an implementation of the proposed system myself. No single existing application met the requirements. That is why my proposed implementation was chosen to implement the idea. The proposed implementation included the use of Bing Search API, AlchemyAPI and CrunchBase API. These APIs were used, because the APIs had better capabilities than my implementation of search engine or NER would have had.

The case study confirmed the concept behind the prototype application was successful. The prototype application produced a result graph according to the case study user request. The result graph visualized connections between technology conferences and their speaker and sponsors. 82% of technology conferences shown in the result graph matched the requirements made by the user. The prototype application recognized speakers and sponsors from the technology conference websites quite well. However the accuracy of the prototype application left room for improvement, because the variance in the recognition accuracy was high. The result graph succeeded to create an ecosystem picture of the technology conferences.

The accuracy of the prototype application would have been better if generalization requirement would have been dropped in the beginning of the prototype application development. The generalization of the prototype application was left from the requirements, because the problem proved to be more difficult than predicted in the start of this thesis. The most important lesson learned from this thesis was the difficulty of finding the matching and intended named-entities from the websites based on user request.

The results from the case study were promising. The prototype application only scratched the surface, when trying to find a way to search and visualize named-entities based on user request from Web. Further research and prototype application development is recommended. The further research will need lot of effort but the prototype application holds a great potential to improve the search experience.

# REFERENCES

"4.11 Scripting — HTML5." http://www.w3.org/TR/html5/scripting-1.html#the-canvas-element (May 19, 2015).

"8.1. What Is Cypher? - - The Neo4j Manual v2.2.6." http://neo4j.com/docs/stable/cypher-introduction.html (October 20, 2015).

"AlchemyAPI ." http://www.alchemyapi.com/ (June 1, 2015).

"Alexa Top 500 Global Sites." http://www.alexa.com/topsites (October 3, 2015).

Alexis Goldstein, Louis Lazaris, Estelle Weyl. 2015. *HTML5 & CSS3 For The Real World*. 2nd Editio.

"Bing Search API." https://datamarket.azure.com/dataset/bing/search (June 1, 2015).

Brin, Sergey, and Lawrence Page. 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine BT - Computer Networks and ISDN Systems." 30: 107–17.

Campbell, William M, Charlie K Dagli, and Clifford J Weinstein. 2013. "Social Network Analysis with Content and Graphs." 20(1).

"CascadiaFest 2015." http://2015.cascadiajs.com/ (October 20, 2015).

Crowston, Kevin, and Marie Williams. 1997. "Reproduced and Emergent Genres of Communication on the World-Wide Web." *Proceedings of the Hawaii International Conference on System Sciences* 6: 30–39.

"CrunchBase." https://www.crunchbase.com/#/home/index (October 3, 2015).

"Custom Search JSON/Atom API." https://developers.google.com/custom-search/json-api/v1/overview#pricing (June 1, 2015).

"Cytoscape.js." http://js.cytoscape.org/ (May 29, 2015).

"Cytoscape.js LGPL-License." https://github.com/cytoscape/cytoscape.js/blob/master/LGPL-LICENSE.txt (May 29, 2015).

"D3.js - Data-Driven Documents." http://d3js.org/ (May 29, 2015).

Donato, Debora, Luigi Laura, Stefano Leonardi, and Stefano Millozzi. 2007. "The Web as a Graph." *ACM Transactions on Internet Technology* 7(1): 4 – es.

Dong, Lei, Carolyn Watters, Jack Duffy, and Michael Shepherd. 2008. "An Examination of Genre Attributes for Web Page Classification." *Proceedings of the Annual Hawaii International Conference on System Sciences*: 1–10.

"DuckDuckGo." https://duckduckgo.com/ (June 1, 2015).

Ecma International. 2013. "ECMA-404: The JSON Data Interchange Format (1st

Edition).” (October).

Eissen, Sven Meyer zu, and Benno Stein. 2004. “Genre Classification of Web Pages.”

Fielding, R.T., and R.N. Taylor. 2000. “Principled Design of the Modern Web Architecture.” *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium*: 407–16.

“Gephi - The Open Graph Viz Platform.” http://gephi.github.io/ (May 29, 2015).

“GNU General Public License v.3.” http://www.gnu.org/licenses/gpl.html (May 29, 2015).

“Google.” https://www.google.com.

Harary, Frank. 1969. “Graph Theory.” : 274.

Haverbeke, Marijn. 2014. *Eloquent JavaScript*. second edi.

Heinonen, Oskari, Kimmo Hätonen, and Mika Klemettinen. 1996. “WWW Robots and Search Engines.” 26(Teollisuuskatu 23): 1–9.

“Import.io.” https://import.io/ (June 1, 2015).

“Introducing the Knowledge Graph: Things, Not Strings.” https://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html (October 20, 2015).

“Introduction to MongoDB — MongoDB Manual 3.0.” http://docs.mongodb.org/manual/core/introduction/ (August 28, 2015).

“JSConf EU 2015.” http://2015.jsconf.eu/ (October 3, 2015).

“JSConf US 2014.” http://2014.jsconf.us/ (October 20, 2015).

“JSConf US 2015 - The Best Conference for JS and the Web. Period.” http://2015.jsconf.us/ (October 18, 2015).

“Kimono.” https://www.kimonolabs.com/ (June 1, 2015).

Lai, Wei, Communication Technologies, and Xiaodi Huang. 2010. “From Graph Data Extraction to Graph Layout : Web Information Visualization.” : 224–29.

Leavitt, Neal. 2010. “Will NoSql Live to Their Promise ?” *IEEE Computer*: 12–14.

“Linkurious.js Graph Visualization Library.” http://linkurio.us/toolkit/ (May 29, 2015).

“Meet Bing, Microsoft’s New Search Engine.” http://searchengineland.com/meet-bing-microsofts-new-search-engine-20093 (October 3, 2015).

“Microsoft and Yahoo Seal Web Deal.” http://news.bbc.co.uk/2/hi/business/8174763.stm (October 3, 2015).

“MOT Oxford Dictionary of English.” https://www.sanakirja.fi/.

"MSN Search Bot a Glimpse of Ambitions." http://www.cnet.com/news/msn-search-bot-a-glimpse-of-ambitions/ (October 3, 2015).

Nadeau, David, and Satoshi Sekine. 2006. "A Survey of Named Entity Recognition and Classification." (1991): 1–20.

"Ng-Conf May 4th - 6th 2016." http://www.ng-conf.org/ (October 20, 2015).

"Nodevember." http://nodevember.org/ (October 20, 2015).

"Nokia." http://www.nokia.com/fi_fi (October 3, 2015).

"Nordic.js." http://nordicjs.com/ (October 20, 2015).

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. "The PageRank Citation Ranking: Bringing Order to the Web." *World Wide Web Internet And Web Information Systems* 54: 1–17.

Purchase, H.C., C. Pilcher, and B. Plimmer. 2012. "Graph Drawing Aesthetics — Created by Users Not Algorithms." 18(1): 81–92.

"Semantria | API." https://semantria.com/api (June 1, 2015).

"Sigma Js." http://sigmajs.org/ (May 29, 2015).

"Sigma.js MIT License." https://github.com/jacomyal/sigma.js/blob/master/LICENSE.txt (May 29, 2015).

"StatCounter Global Stats." http://gs.statcounter.com/press/yahoo-growth-stalls (June 1, 2015).

Sullivan, Dan. 2015. *NoSQL for Mere Mortals*.

"Tampere University of Technology." http://www.tut.fi/en/home (October 3, 2015).

"Text Analytics from Saplo." http://saplo.com/ (June 1, 2015).

"TextRazor." https://www.textrazor.com/ (June 1, 2015).

"The BSD 3-Clause License ." http://opensource.org/licenses/BSD-3-Clause (May 29, 2015).

"Thomson Reuters | Open Calais." http://new.opencalais.com/ (June 1, 2015).

"W3C SVG Working Group." http://www.w3.org/Graphics/SVG/ (May 29, 2015).

"WebGL - OpenGL ES 2.0 for the Web." https://www.khronos.org/webgl/ (May 29, 2015).

"Yahoo BOSS – Pricing." https://policies.yahoo.com/us/en/yahoo/terms/product-atos/boss/pricing/index.htm (June 1, 2015).

"Yahoo: An 18-Year Timeline of Events | PerformanceIN." http://performancein.com/news/2012/07/17/yahoo-18-year-timeline-events/

(October 3, 2015).

Yates, Joanne, and Wanda J. Orlikowski. 1992. “Genres of Organizational Communication: A Structurational Approach to Studying Communication and Media.” *The Academy of Management Review* 17(2): 299.

# APPENDIX

| Supposed conference website | Conference website |
|---|---|
| 2011.ffconf.org | Yes |
| html5live.org | Yes |
| html5devconf.com | Yes |
| 2011.texasjavascript.com | Yes |
| 2013.texasjavascript.com | Yes |
| sol.lp.findlaw.com | No |
| 2013.lxjs.org | Yes |
| html5tx.com | Yes |
| 2009.ffconf.org | Yes |
| jsconfbp.com | Yes |
| chicagowebconf.org | Yes |
| ng-conf.org | Yes |
| jqueryuk.com | Yes |
| webdesignworld.com | No |
| 2012.jsday.it | Yes |
| ongamestart.com | No |
| 2011.buildconf.com | Yes |
| 2012.buildconf.com | Yes |
| 2013.buildconf.com | Yes |
| minnewebcon.org | Yes |
| jsnext.net | Yes |
| smartwebconf.com | Yes |
| onandroidconf.com | Yes |
| 2015.cascadiajs.com | Yes |
| nordicjs.com | Yes |
| acconference.com | No |
| 2010.jsconf.us | Yes |
| 2014.cascadiajs.com | Yes |
| scotlandjs.com | Yes |
| artifactconf.com | Yes |
| 2013.ffconf.org | Yes |
| meetup.com | No |
| ajaxian.com | No |
| rejectjs.org | Yes |
| jsilconf.com | Yes |
| 2014.jsconf.asia | Yes |
| spacecityjs.com | Yes |
| connect-js.com | Yes |
| nodepdx.org | Yes |
| events.jquery.org | No |
| nodedublin.com | Yes |

| | |
|---|---|
| nodesummit.com | Yes |
| nodevember.org | Yes |
| wdcnz.com | Yes |
| techcrunch.com | No |
| jsist.org | Yes |
| jdc2013.egjug.org | No |
| developerweek.com | Yes |
| 2012.spainjs.org | Yes |
| codemesh.io | Yes |
| midwestjs.com | Yes |
| javascript-conference.de | Yes |
| iloveapis.com | Yes |
| gluecon.com | Yes |
| venturebeat.com | No |
| 2015.jsconf.eu | Yes |
| devday.pl | Yes |
| bbconference.com | No |
| 2013.jsconf.eu | Yes |
| 2014.jsconf.us | Yes |
| 2014.jsconf.eu | Yes |
| 2015.jsconf.us | Yes |